



Outils d'aide à la conception de circuits numériques

Manuel d'utilisation



Département TIC
Institut REDS
Septembre 2008, version 6.0



Date	Version	Ingénieur	Révision
23/09/08	v6.0	YNG	Passage à Latex et aux outils EDA suivants : HDL Designer 2007.1, ModelSim6.2d, Precision 2007a_8, Quartus 7.2

Table 1 – Révisions

Auteur et version du manuel Les premières versions de ce manuel ont été écrites par Michel Salamin. Cette version est une mise à jour en prenant compte des modifications qui sont apparues avec les nouvelles versions des outils.

Mise à jour de ce manuel Je remercie tous les utilisateurs de ce mode d'emploi de m'indiquer les erreurs qu'il comporte, ainsi que les problèmes qui apparaissent avec les logiciels de «Mentor Graphics» en suivant les procédures indiquées dans ce manuel. De même, si des informations semblent manquer ou sont incomplètes, elles peuvent m'être transmises, cela permettra une mise à jour régulière de ce manuel.

Contact

Auteur	Jean-Pierre Miceli
Resp	Graf Yoan
e-mail	yoan.graf@heig-vd.ch
Tel	+41 (0)24 / 55 76 259
Adresse	Institut REDS, Reconfigurable & Embedded Digital Systems Heig-vd, Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud Route de Cheseaux 1 CH-1400 Yverdon-les-Bains
Tel	+41 (0)24 / 55 76 330 (central)
Fax	+41 (0)24 / 55 76 404
E-mail	reds@heig-vd.ch
Internet	http ://reds.heig-vd.ch

Autres personnes à contacter en cas d'absence

Directeur du REDS	Messerli Etienne
Tél. direct	+41 (0)24 55 76 302
e-mail	etienne.messerli@heig-vd.ch

Table des matières

1 Introduction

Ce manuel est un introduction aux logiciels EDA (Electronic Desing Automation). Il n'a pas la prétention de couvrir tous les aspects de ceux-ci. Ce manuel a pour objectif de permettre aux étudiants de saisir le schéma de circuits logiques pour les simuler et les intégrer dans un PLD (Programmable Logic Device, circuit logique programmable) . Il a été écrit afin de permettre une approche simple et efficace du logiciel. Ce document comporte des indications spécifiques liées à la configuration des logiciels utilisés au sein de l'institut REDS de la HEIG-VD. Cependant, il ne se limite pas à résumer le mode d'emploi de divers logiciels, mais introduit une méthode de travail permettant la réalisation complète de circuits numériques avec le langage VHDL.

Ce manuel permet, d'autre part, l'utilisation d'outils modernes pour la conception de circuits numériques (logiciels EDA, Electronic Design Automation) associés aux nouvelles méthodologies.

1.1 Outils pour la conception

de circuits La conception de circuits met en œuvre quatre outils : un éditeur de texte/entrée graphique, un simulateur, un synthétiseur et un placeur-routeur. Tous ces outils sont prévus pour l'utilisation du langage VHDL.

1.1.1 HDL Designer (version 2007.1)

HDL Designer est un logiciel de gestion de projets et de saisie graphique utilisé dans la conception de circuits numériques. La saisie graphique est une alternative à la saisie textuelle. Ce logiciel facilite la gestion de projets et les saisies graphiques lors de la réalisation d'un système numérique en vue d'une intégration dans un circuit logique programmable (CPLD ou FPGA). En plus de la fonction d'entrée graphique, il sert à piloter d'autres logiciels comme le simulateur dont vous allez utiliser les services lors des laboratoires. De cette manière, vous n'aurez à connaître qu'un unique logiciel. Pour que cette collaboration fonctionne bien, nous avons paramétré certaines options afin que les logiciels puissent communiquer entre-eux. Cela implique un respect scrupuleux de la mise en route d'un projet, d'où la présente documentation que nous vous demandons de suivre ***à la lettre***.

1.1.2 ModelSim (version SE 6.3d)

Ce logiciel permet de simuler les descriptions réalisées afin de vérifier si elles respectent bien les spécifications.

1.1.3 Precision Synthesis (version 2007a.8)

Ce logiciel effectue la synthèse d'une description VHDL en un schéma logique équivalent. Celui-ci est composé uniquement des fonctions logiques disponibles dans le circuit cible choisi. Il s'agit de fonctions logiques simples tel que porte ET, OU, flip-flop, latch, ...

1.1.4 Quartus II (version 7.2)

Ce logiciel effectue le placement-routage dans le circuit cible spécifique : il attribue des portions (cellules) du circuit programmable aux fonctions identifiées lors de la synthèse, planifie les interconnections et génère le fichier de programmation (configuration) du circuit cible choisi.

Parmi ces outils, le placeur-routeur est le seul qui soit destiné spécifiquement à des circuits d'un certain fabriquant. Il est fourni exclusivement par celui-ci, en l'occurrence la société *Altera*.

1.2 Exemple "Majorité"

Pour illustrer nos propos, nous allons créer tout au long de ce tutorial un projet simple, dont vous pourrez reprendre les principes de construction pour les étendre à vos besoins.

Nous allons réaliser un système permettant de déterminer s'il y a une majorité sur trois votants.

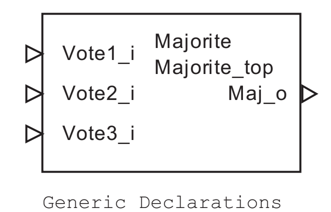


FIGURE 1.1 – Symbole du projet "Majorité" utilisé comme exemple

Vote3	Vote2	Vote1	Maj
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

TABLE 1.1 – Table de vérité

L'équation résultante, après simplification :

$$Maj = Vote2 \cdot Vote1 \# Vote3 \cdot Vote1 \# Vote3 \cdot Vote2$$

1.3 Etapes de conception

La conception d'un système passe par quatre étapes principales :

1. Réflexion et conception
2. Réalisation
3. Vérification
4. Intégration

La première étape sort du cadre de ce manuel. Elle fait partie du cours théorique de système numérique. Les étapes suivantes seront présentées à l'aide de l'exemple. Elles nous permettront de passer de la solution conçue à une intégration dans un circuit.

La deuxième étape consiste à saisir votre solution dans le logiciel *HDL Designer*.

Pour la troisième étape vous pourrez réaliser une simulation manuelle en appliquant des valeurs sur les entrées et en vérifiant visuellement l'état des sorties. Dans un deuxième temps, vous disposerez d'un banc de test automatique pour une vérification complète de votre solution.

La dernière étape vous permettra de réaliser la traduction de votre schéma en vue de son intégration dans un circuit logique programmable (CPLD ou FPGA). Il s'agit d'une synthèse suivie par le placement-routage.

2 Présentation de HDL Designer

Ce chapitre donne les informations indispensable pour l'utilisation de *HDL Designer*.

2.1 Organisation du répertoire d'un projet

Le répertoire d'un projet contient différents répertoires et fichiers décris ci-après. Les projets que nous allons réalisés contiennent toujours au minimum une bibliothèques :

– Bibliothèque de travail, pour la réalisation du travail demandé

Les bibliothèques de base, *REDS_Lib_Base* et *REDS_Lib_IO*, sont des bibliothèques partagées. Elles sont toujours disponibles et se situent sur le disque *C :*.

Dans le cas de projets plus important, il est possible d'avoir des bibliothèques supplémentaires.

Le nom du répertoire du projet est de la forme *nomProjet_proj*. Le sufixe _proj est obligatoire. Le chemin d'accès à ce dermier doit correspondre à :

D :\Classe\nom_groupe\nomProjet_proj

Dans ce répertoire se trouve :

– Les dossiers correspondants à chaque bibliothèque composant le projet. Chacun de ces répértoires est constitué des dossiers décrits au paragraphe 2.1.1.
– Le fichier *nomProjet.hdp*. Il contient les chemins d'accès des différentes bibliothèques utilisées dans le projet.

***Remarque :* Pour les noms des différents répertoires, il ne faut utiliser que les caractères non accentués (de 'A' à 'Z') et "l'underline" ('_'). Il ne faut absolument pas utiliser d'espace ou de signe tel que le '&' !**

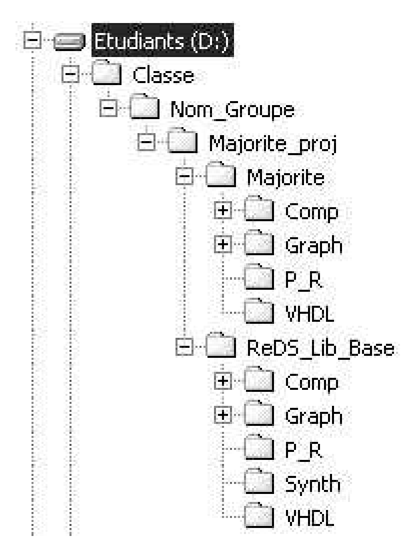


FIGURE 2.1 – Organisation d'un répertoire de travail

2.1.1 Structure du répértoire d'une bibliothèque

2.1.1.1 Répertoire "Comp"

Ce répertoire contient les fichiers de sortie du compilateur de *ModelSim*. Ces fichiers permettent de simuler les différents modules du projet.

2.1.1.2 Répertoire "Graph"

Ce répertoire contient les données graphiques, créés par *HDL Designer*, servant à mémoriser les différents symboles, graphes d'états, organigrammes, tables de vérité.

2.1.1.3 Répertoire "P_R"

Ce répertoire contient les fichiers d'entrée et de sortie, pour chaque module, du logiciel de placement-routage (logiciel *Quartus II*).

2.1.1.4 Répertoire "Synth"

Ce répertoire contient les fichiers VHDL d'entrée et de sortie, pour chaque module, du logiciel de synthèse (logiciel *Precision Synthesis*).

2.1.1.5 Répertoire "VHDL"

Ce répertoire contient les fichiers sources VHDL (non graphiques) et les fichiers générés par *HDL Designer*. Les fichiers générés sont la traduction, en VHDL, des différents objets graphiques se trouvant dans le répertoire *"Graph"*. Ces fichiers seront compilés avec le compilateur de *ModelSim* en vue de la simulation ou avec le compilateur *"Generic 1 file"* de *HDL Designer* en vue de la synthèse, généralement, avec *Precision Synthesis*.

2.2 Création d'un nouveau projet

Un projet est constitué d'un dossier ayant le nom *[Nom_Projet]_proj* contenant différents dossiers et un fichier nommé *[nom_projet].hdp*. Dans le cadre du projet *Majorité*, le dossier projet se nomme *majorite_proj* dans lequel on trouve le dossier majorite et le fichier *Majorite.hdp*.

2.2.1 Création du fichier nomProjet.hdp

La création d'un projet est automatisée à l'aide du script *HDL_NewProject_MultiLibrary.tcl*, disponible dans le menu : *Démarrer* ▷ *labo numérique* ▷ *HDL_NewProject_MultiLibrary.tcl* Cet utilitaire permet de créer le fichier projet (**.hdp*) utilisé par *HDL Designer*. Ce fichier contient le mapping des différentes bibliothèques contenues dans le projet.

2.2.1.1 Utilisation de HDL_NewProject_MultiLibrary.tcl

1. Créer, s'il n'existe pas encore, le répertoire du projet (*NomProjet_proj*).
2. Lancer le script *HDL_NewProject_MultiLibrary.tcl*, la fenêtre de la figure 2.2 apparaît.
3. Séléctionner le répertoire du projet à l'aide du bouton *Choisir*.
4. Ajouter les bibliothèques désirées, soit :
   (a) Sous *Bibliothèques standards*, séléctionnez *NomProjet* pour créer une bibliothèque portant le nom *NomProjet*. Un composant *Top_Sim* sera automatiquement inclu dans cette bibliothèque (nous verrons plus tard son utilité).
   (b) Si nécessaire sous *Bibliothèques spécifiques*, donnez le nom d'autres bibliothèques pour le projet.
   (c) Les bibliothèques standards *REDS_Lib_Base* et *REDS_Lib_IO* sont automatiquement intégrées.

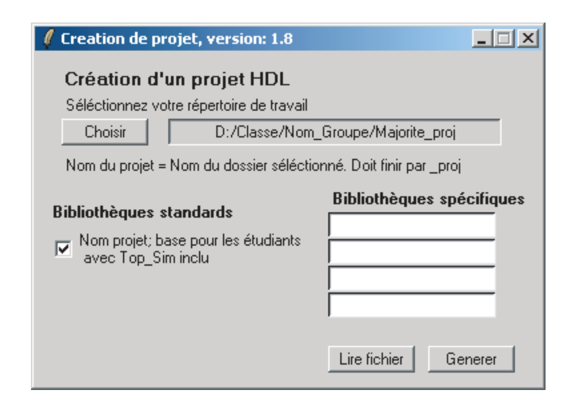


FIGURE 2.2 – Fenêtre de création d'un nouveau projet

Cet utilitaire crée le fichier *NomProjet.hdp* contenant le mapping des différentes bibliothèques. Il crée aussi les répertoires des différentes bibliothèques en y incluant les répertoires *P_R* et *Graph*.

Les autres répertoires soit *Comp*, *Synth* et *VHDL* seront créés automatiquement par *HDL Designer* lors de l'utilisation des *Tasks*, par exemple le répertoire *Comp* sera créé lors de la première utilisation du *Task Compile*.

2.2.2 Ajout d'une bibliothèque à projet existant

1. Lancer le script *HDL_NewProject_MultiLibrary.tcl*, la fenêtre de la figure 2.2 apparaît.
2. Séléctionner le répertoire du projet à l'aide du bouton *choisir*.
3. Relecture du fichier projet en cliquant sur le bouton *Lire fichier*.
4. Ajouter les bibliothèques désirées à celle déjà disponible, soit :
   (a) Sous *Bibliothèques standards*, séléctionnez *NomProjet* pour créer une bibliothèque portant le nom *NomProjet*.
   (b) Sous *Bibliothèques spécifiques*, donnez le nom d'autres bibliothèques pour le projet.

2.3 Lancement de HDL Designer et ouverture du projet

Un projet est en général soit fourni par votre professeur (en général à l'adresse *eint20**reds**labo*\...) soit une sauvegarde personelle d'un projet en cours. **Copier les sources de votre projet, le répertoire *[nom_projet]_proj*, dans votre répertoire de travail (*D :**classe**groupe*\).**

Dans le cas du projet *Majorité*, il faut copier le dossier *Majorite_proj* complet à l'adresse *D :**classe**groupe*\

2.3.1 Lancement du programme

Pour lancer le programme *HDL Designer*, allez dans :

Demarrer ▷ Labo Numerique ▷ EDA ▷ *HDL Designer 2007.1*

Une fois *HDL Designer* lancé, vous vous trouvez dans le *Projet Manager* comme montre a la figure 2.3.

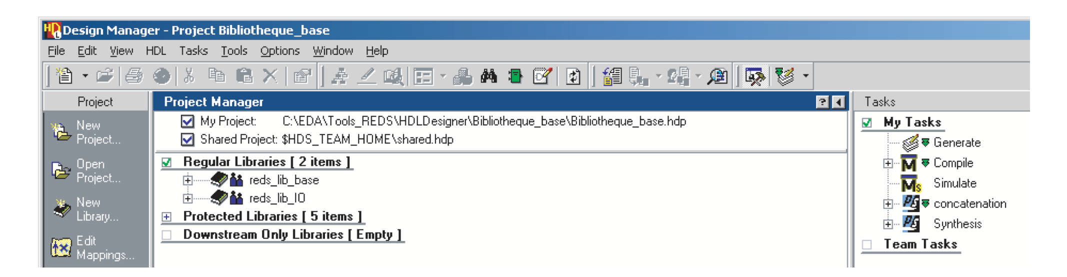


FIGURE 2.3 – Design Manager suite au lancement de *HDL Designer*

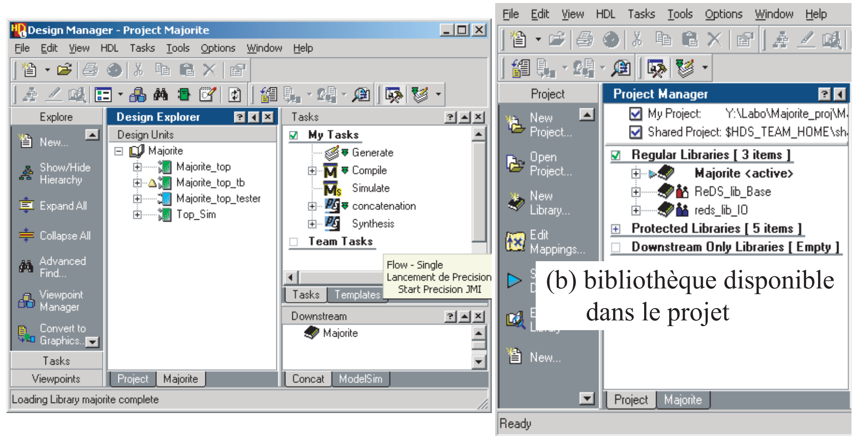
2.3.2 Chargement d'un projet

Opérations à suivre pour charger un projet :

- Bouton *Open Projet* (partie de gauche du *Design Manager*)
- Dans la fenêtre qui apparaît, bouton *Browse*
- Sélectionner le fichier *nom_du_projet.hdp* situé dans le répertoire [*Nom_Projet*]*_proj* (dans le cas du projet *Majorité*, le chemin est *D :**Classe**Groupe**Majorite_proj**majorite.hdp*) et cliquez sur *Ouvrir*
- Cliquez sur OK

HDL Designer ouvre le projet et vous devriez vous trouvez dans une situation similaire à la figure 2.4. La bibliothèque *Majorité* a été ouverte automatiquement.

Dans le *Projet Manager*, on voit les bibliothèques disponibles dans le projet.



(b) bibliothèque disponible dans le projet

(a) situation lors de l'ouveture d'un projet

(c) onglet de navigation entre projet browser et les bibliothèque

Figure 2.4 – Ouverture du projet Majorité (bibliothèque vide)

Pour la réalisation et la simulation de projet, le REDS met à disposition deux bibliothèques :

- *REDS_Lib_Base*
- *REDS_Lib_IO*

Celles-ci seront détaillées par la suite au travers d'exemples.

2.4 Création du composant top

Lors d'une *description hiérarchique*, le composant de niveau le plus haut se nomme toujours *nom-Projet_top* (dans notre example *majorite_top*).

2.4.1 Création du symbol top

Allez dans la barre verticale à gauche de l'écran et sélectionnez dans l'onglet *"Main"* ▷ *"New/Add"*. La fenêtre de la figure 3.1 apparait.

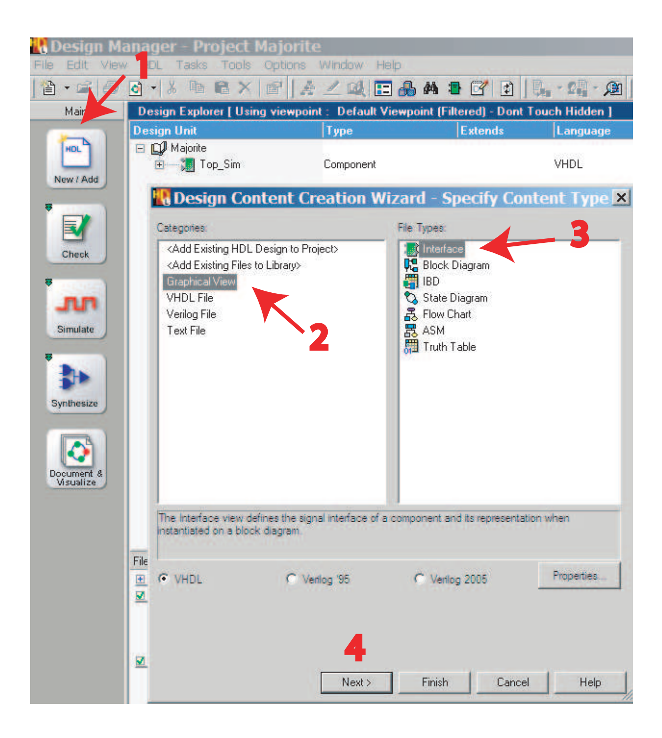


FIGURE 2.5 – Création d'un nouveau symbole

Sélectionnez ▷ *"Graphical View"* ▷ *"Interface"* ▷ *"Next>"*. La fenêtre de la figure 3.2 apparaît.

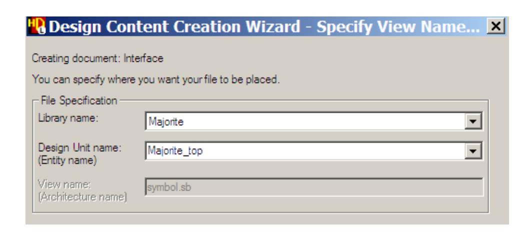


FIGURE 2.6 – Définition du nom du symbole

Lors de l'enregistement d'un symbole, il faut spécifier la librairie dans laquelle fait partie ce composant (dans le champ *Library name*).

Le symbole enregistré étant le composant principal du projet (le *top*), il faut lui donner comme nom (dans le champ *Design Unit Name*) le nom du projet avec l'adjonction du suffixe "*_Top*" ce qui donne, dans notre exemple *Majorite_Top*. Puis cliquez sur *Finish*. La fenêtre symbol(Interface) (figure 3.3, fenêtre du haut) s'ouvre.

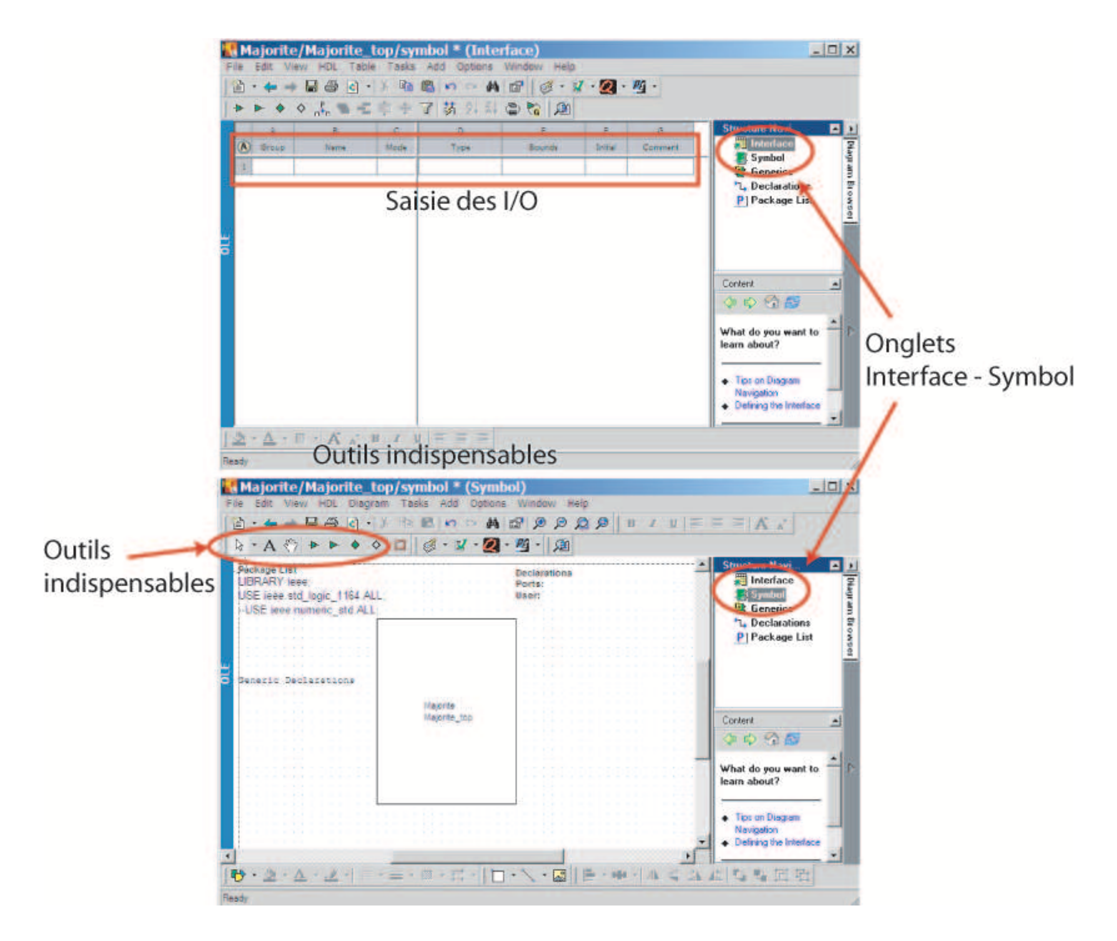


Figure 2.7 – Création/Edition d'un symbole

Deux méthodes de saisie des entrées/sorties sont disponibles. La première est textuelle et la seconde est graphique et permet de placer les entrées/sorites autour du symbole.

On remarque dans ces fenêtres les différents points importants (cf. figure 2.7) :

– Les outils indispensables
– Les deux onglets Interface et Symbol

2.4.1.1 Outils indispensables

	L'outil de sélection
A	Loutil d'ajout de commentaires
	L'outil d'ajout de port d'entrée à l'entité
	L'outil d'ajout de port de sortie à l'entité
	L'outil d'ajout de port d'entrée/sortie à l'entité

Table 2.1 – Outils indispensables

2.4.1.2 Ajout de port

Pour ajouter des *"ports"*, dans la vue graphique *Symbol(Symbol)*, sélectionnez un des outils d'ajout (entrée, sortie, . . .) puis ajouter autant de signaux que nécessaire en cliquant sur les bords du symbole à l'endroit souhaité. Pour revenir à l'outil de sélection, il suffit de cliquer avec le bouton droit de la souris ou appuyer sur la touche ESC.

2.4.1.3 Définition des propriétés de chaque "port"

Pour définir les propriétés des *ports* ; soit son nom, son type ainsi que la taille, allez dans l'onglet *Interface* ou faire un double-clic sur un triangle représentant un port. Dans la fenêtre qui apparaît, entrez les valeurs afin d'obtenir le même résultat que celui de la fenêtre de la figure 2.8.

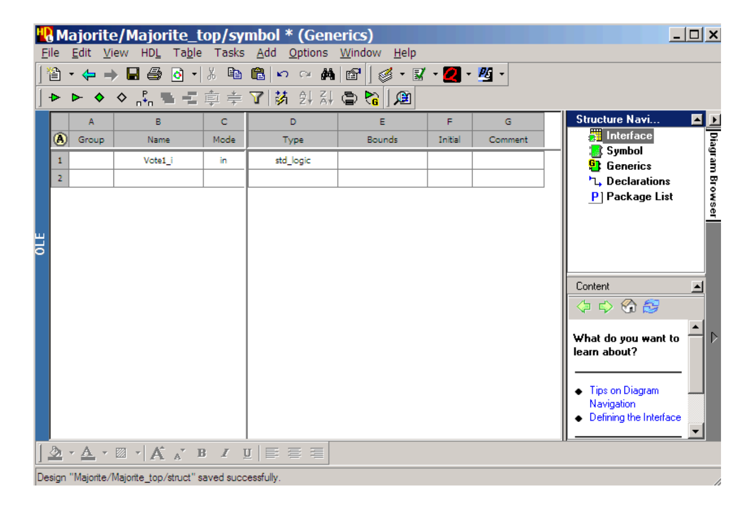


FIGURE 2.8 – Définition des propriétés des ports du composant pour le projet *"Majorité"*

2.4.2 Création du schéma du top

Pour la création du schéma du top :

– Faite un double-clic sur le composant créé au point précédent. La fenêtre de la figure 2.9 apparaît

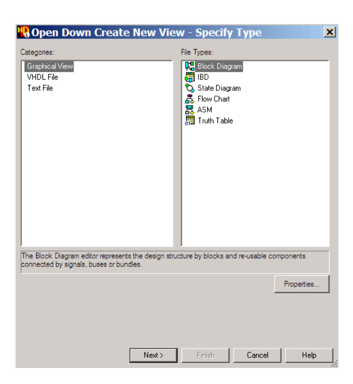


FIGURE 2.9 – Fenêtre de choix du type de schéma (architecture) d'un composant

– Sélectionnez *Graphical View* ▷ *Bloc Diagram* et cliquez sur le bouton *Next >*
– Dans la nouvelle fenêtre, validez avec le bouton *Finish*
– Vous vous retrouvez dans la même situation que montrée à la figure 2.10. On retrouve les signaux créés au point précédent

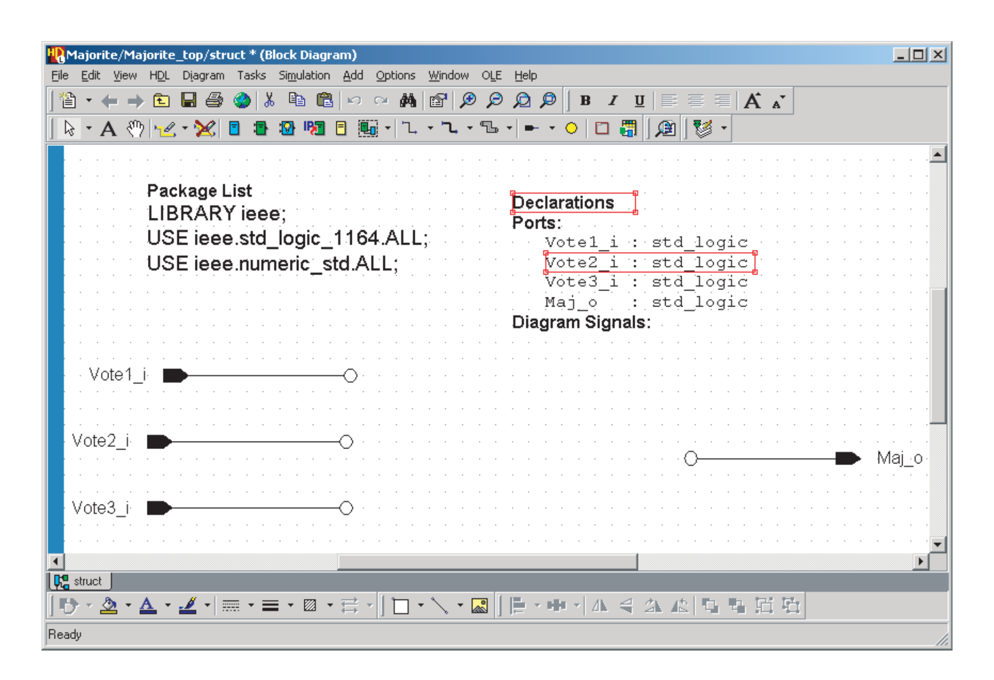


FIGURE 2.10 – Situation suite à la création d'un schéma

2.5 Sauvegarde d'un projet

Le disque dur local "D :" peut être effacé d'une séance à l'autre sans préavis !

A la fin de chaque séance, chaque groupe doit faire une copie du dossier contenant le projet sur son compte personnel (qui se trouve généralement sous le lecteur réseau "T :" et sur un autre support (clé USB, ou autres).

Chaque groupe doit disposer d'au minimum 3 sauvegardes. Chaque étudiant doit avoir accès à au moins une copie du projet.

De par la limitation en taille des comptes personnels à la HEIG-VD, il est souvent nécessaire de ne garder que le minimum d'information utile afin de pouvoir sauvegarder tous les travaux de laboratoire en cours.

En ce qui concerne le laboratoire de systèmes numériques, il est possible d'alléger la taille des données à sauvegarder en supprimant les sous-dossier du projet suivants :

– Dans le dossier des blibliothèques qui se trouve à l'intérieur du dossier projet :
– *Comp*
– *Synth*
– *P_R*

La suppression de ces dossiers nécessitera d'effectuer, à nouveau, les opérations suivantes :

1. Création du répértoire *P_R* dans les dossiers des bilibothèques
2. Génération des descriptions VHDL
3. Compilation des descriptions VHDL
4. Synthèse Placement-routage

Si la place sur votre compte personnel le permet, il est conseillé de ne pas supprimer les répertoires *synth* et *P_R*. Vous supprimez uniquement le répertoire *Comp*.

2.6 Saisie de schémas, description du projet

2.6.1 Ouverture du schéma du top

Pour ouvrir le composant *_top* (dans notre exemple *Majorite_top*) :

1. Sélectionnez la bibliothèque de travail à l'aide des onglets
2. Double-cliquez le composant (dans notre exemple *Majorite_top*)
3. Le composant s'ouvre dans une nouvelle fenêtre (affichage de son architecture)

2.6.2 Saisie d'un schéma

2.6.2.1 Outils indispensables

	L'outil d'ajout de blocs
	L'outil d'ajout de composants
	L'outil d'ajout d'expressions concurrentes
	L'outil d'ajout de signaux à un élément
	L'outil d'ajout de vecteurs (signaux multi-éléments)
	L'outil d'ajout de commentaires

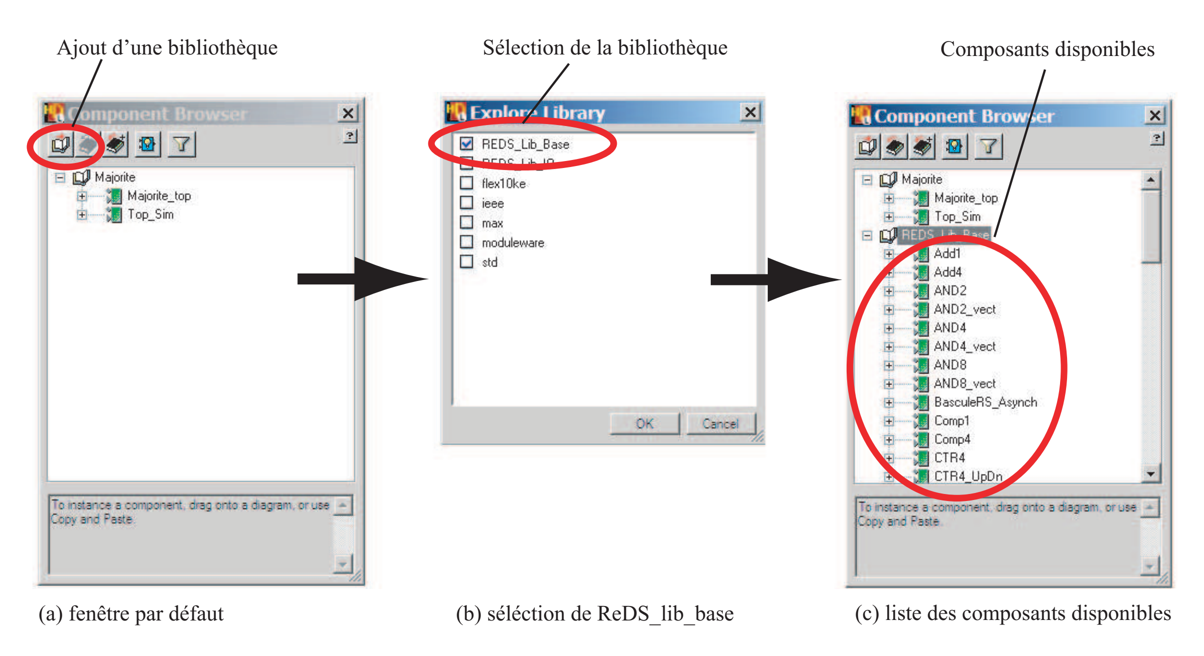
TABLE 2.2 – Outils indispensables

2.6.2.2 Importation de composant de REDS_Lib_Base

Pour notre projet *Majorité*, nous avons besoin de trois portes AND2, une porte OR4 et un GND. Tous ces composants sont disponibles dans *REDS_Lib_Base*.

Pour les inclure dans notre schéma :

1. Ouvrir la fenêtre *Componant Browser* en utilisant l'outil d'ajout de composants (, *Add component*).
2. Dans la nouvelle fenêtre, sélectionnez la bibliothèque *REDS_Lib_Base* comme montré à la figure 2.11.
3. Glissez-Déposez les composants nécessaires dans votre schéma



(a) fenêtre par défaut (b) séléction de ReDS_lib_base (c) liste des composants disponibles

FIGURE 2.11 – Sélection de la bibliothèque ReDS_lib_base

2.6.2.3 Interconnexion

Les interconnexions se font très facilement avec l'outil *Ajout de signal* ou *Ajout de vecteur*. Un vecteur sera utilisé pour interconnecter des éléments ayants un port de plusieurs bits de large, alors

qu'un signal sert à interconnecter un port d'un seul bit. En pointant directement sur les ports des éléments, les vecteurs prennent directement le bon nombre de bits.

Les signaux prennent les noms des entrées/sorties (ports*) sur lesquelles nous les avons connectés en premier. Pour changer ce nom, on peut double-cliquer sur le signal et le changer dans la boîte de dialogue qui apparaît sous l'étiquette *Name*. Par convention, tous les signaux internes à un composant prennent le suffixe : "*_s*". On peut "couder" un signal en cliquant une fois sur son trajet.

****N.B***. : dans la suite de ce manuel, on utilisera le terme anglais port pour désigner indifféremment une entrée ou une sortie.

2.6.2.4 Schéma bloc résultant

Après l'importation et l'interconnexion, nous obtenons le schéma de la figure 2.12.

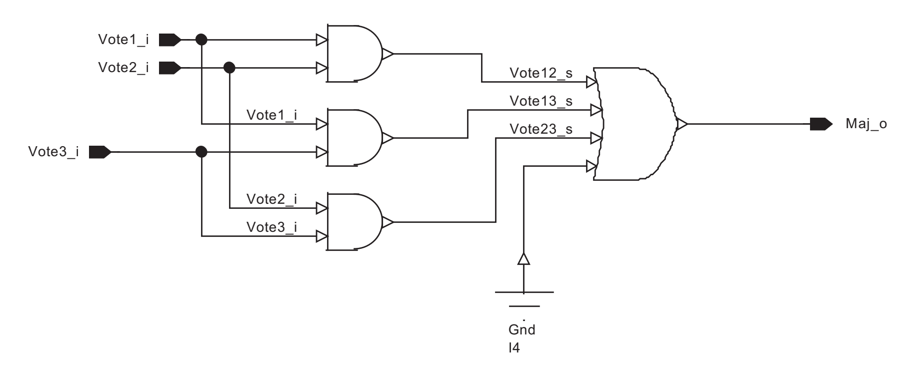


FIGURE 2.12 – Schéma bloc du projet *Majorité*

2.6.3 Exportation du schéma réalisé

Afin d'agrémenter votre rapport de jolies illustrations, il est possible de coller le schéma directement dans Word ou un autre logiciel utilisant la technologie OLE. Tout objet graphique peut-être exporté en effectuant un cliquer-déplacer de la barre bleue qui se trouve à gauche de la fenêtre vers la fenêtre du document de réception.

2.6.4 Impression

HDL Designer permet d'agencer directement la mise en page de toute la conception. Vous pouvez définir le nombre de pages sur lesquelles le document sera imprimé, *HDL Designer* mettra les éléments à la bonne échelle.

En allant dans le menu *File* choisissez *Page Setup...* Dans cette fenêtre l'option *Fit to* permet de spécifier le nombre de pages. L'option *Show Header* and *Footer* permet de faire figurer la date, l'auteur et le chemin du fichier.

2.7 Concaténation et explosion de vecteurs

Pour l'utilisation de certains composants, il sera nécessaire de regrouper des signaux en vecteurs ou d'exploser un vecteur en plusieurs signaux.

Cette explication sera utilisée dans le cas où le projet fourni contient un vecteur en entrée ou en sortie.

Pour illustrer la marche à suivre, nous allons utiliser un multiplexeur d'un bus de 2 bits avec 2 muliplexeurs 2to1 (disponible dans la bibliothèque *REDS_Lib_Base*).

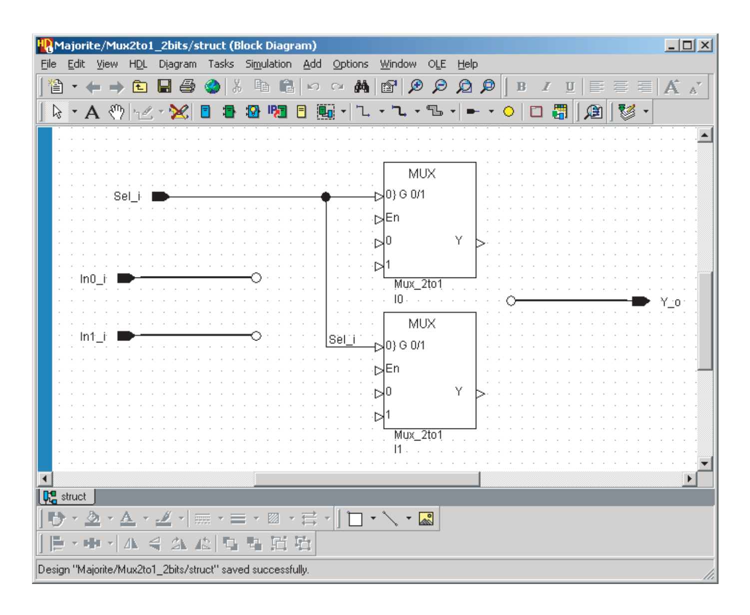


FIGURE 2.13 – Multiplexeur 2to1 avec vecteurs de 2 bits

2.7.1 Connexion d'une partie d'un vecteur

Nous allons extraire des signaux aux vecteurs et vice-versa :

1. Sélectionnez l'outil *Ajout de signal*
2. Rattachez-le au port de la porte qui vous intéresse
3. Allongez le pour le connecter au vecteur.

Lors de la connexion, *HDL Designer* va ouvrir une boîte de dialogue qui vous permettra de choisir l'indice du signal correspondant dans le vecteur. Le signal prendra le nom du vecteur ainsi que l'indice du bit qu'il représente entre parenthèse.

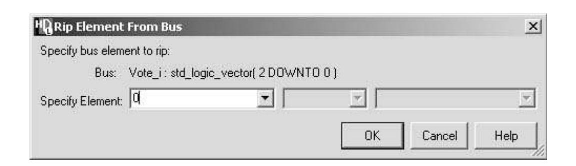


FIGURE 2.14 – Fenêtre de "extraction" de signaux

Si vous vous êtes trompés, la méthode de correction la plus simple consiste à effacer le signal erroné et à refaire l'opération. Si cette méthode ne vous convient pas, vous pouvez ouvrir les propriétés du signal incriminé et les modifier en donnant le même nom que le vecteur (que *HDL Designer* va reconnaître) et compléter le champ *Slice/Index* avec l'index du bit qui vous intéresse. La figure 2.15 illustre le résultat obtenu.

Pour regrouper des sorties distinctes en un vecteur, on procédera de la même manière, sauf qu'il faudra connecter d'abord le signal au vecteur et ensuite le connecter au port.

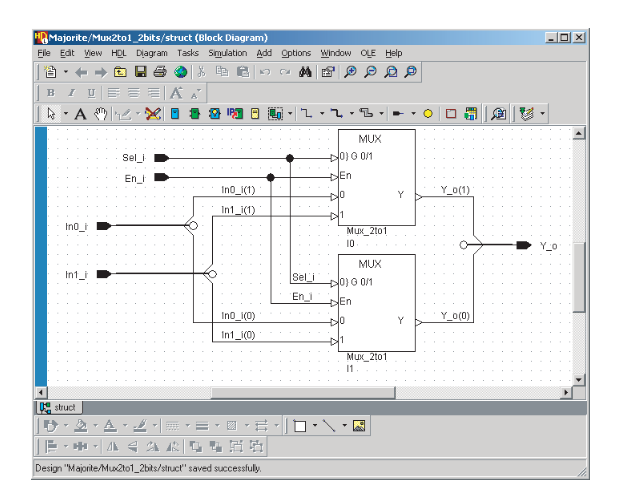


FIGURE 2.15 – Résultat du multiplexeur une fois les connexions effectuées

Utilisation de HDL Designer 3

Ce chapitre donne une description de la mise en oeuvre des fonctions que propose *HDL Designer*. Pour obtenir les informations concernant l'utilisation de *HDL Designer* dans le flow de conception, veuillez vous référer au chapitre 2, "Présentation de *HDL Designer*".

3.1 Création d'un nouveau symbole

Allez dans la barre verticale à gauche de l'écran et sélectionnez dans l'onglet *"Main"* ▷ *"New/Add"*. La fenêtre de la figure 3.1 apparait.

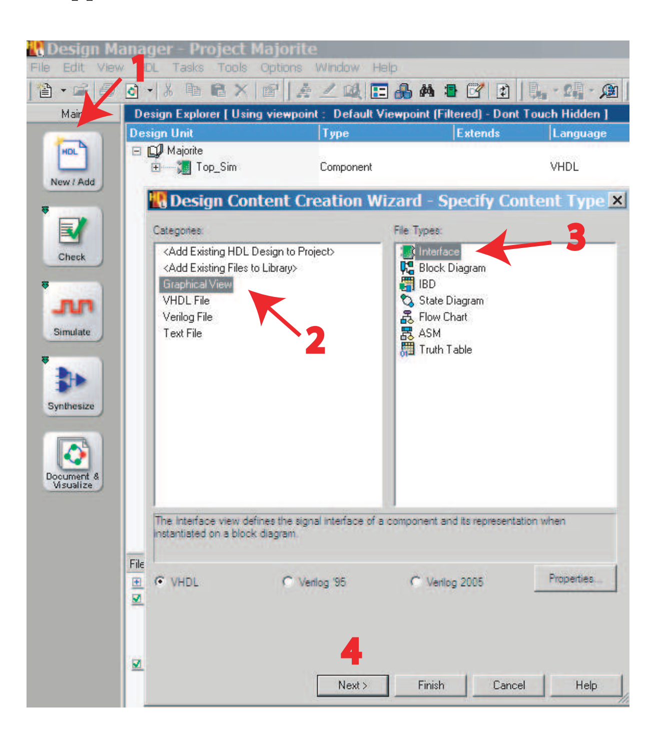


FIGURE 3.1 – Création d'un nouveau symbole

Sélectionnez ▷ *"Graphical View"* ▷ *"Interface"*. La fenêtre de la figure 3.2 apparaît.

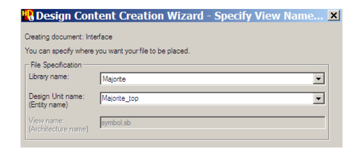


FIGURE 3.2 – Définition du nom du symbole

Lors de l'enregistement d'un symbole, il faut spécifier la librairie dans laquelle fait partie ce composant (dans le champ *Library name*).

Si le symbole enregistré est le composant principal du projet (le *top*), il faut lui donner comme nom (dans le champ *Design Unit*) le nom du projet avec l'adjonction du suffixe *_top* ce qui donne, dans notre exemple *monProjet_top*. Puis cliquez sur *Finish*. La fenêtre symbol(Interface) (figure 3.3, fenêtre du haut) s'ouvre.

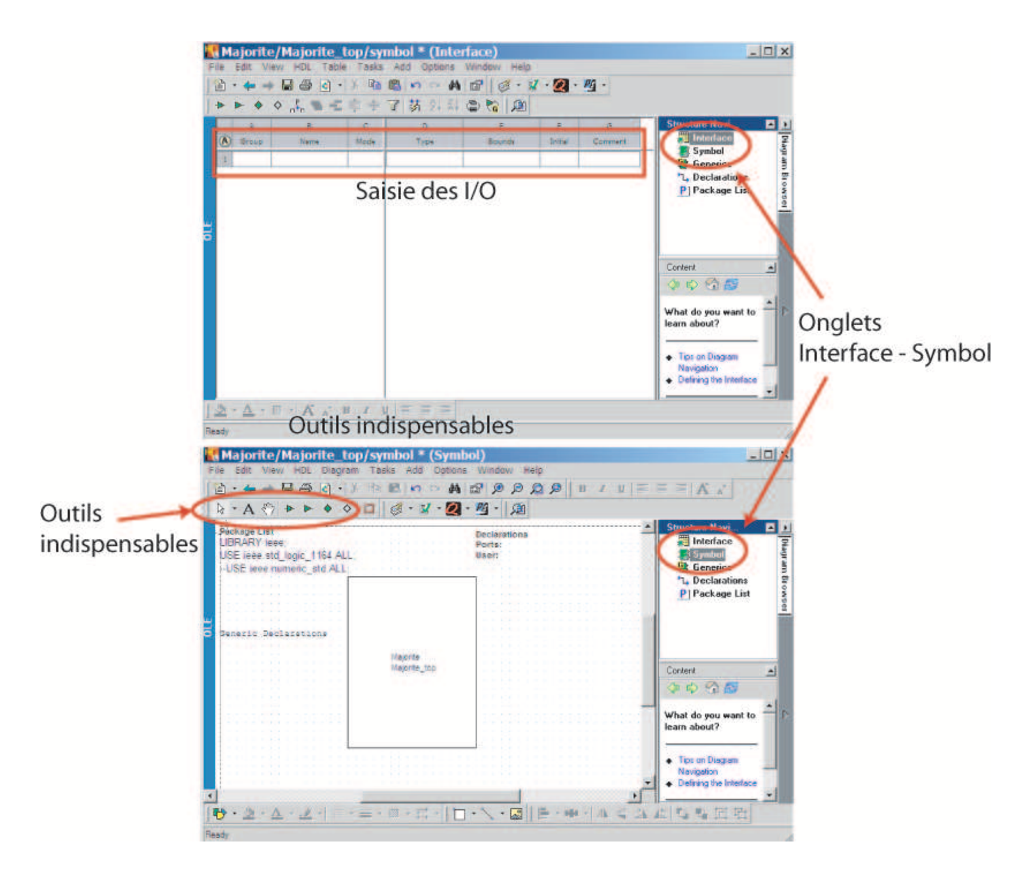


FIGURE 3.3 – Création/Edition d'un symbole

Deux méthodes de saisie des entrées/sorites sont disponibles. La première est textuelle et la seconde est graphique et permet de placer les entrées/sorites autour du symbole.

On remarque dans ces fenêtres les différents points importants (cf. figure 3.3) :

- Les outils indispensables
- Les deux onglets Interface et Symbol

3.1.0.1 Outils indispensables

	L'outil de sélection
A	Loutil d'ajout de commentaires
	L'outil d'ajout de port d'entrée à l'entité
	L'outil d'ajout de port de sortie à l'entité
	L'outil d'ajout de port d'entrée/sortie à l'entité

TABLE 3.1 – Outils indispensables

3.1.0.2 Ajout de port

Pour ajouter des *"ports"*, dans la vue graphique *Symbol(Symbol)*, sélectionnez un des outils d'ajout (entrée, sortie, . . .) puis ajouter autant de signaux que nécessaire en cliquant sur les bords du symbole à l'endroit souhaité. Pour revenir à l'outil de sélection, il suffit de cliquer avec le bouton droit de la souris ou appuyer sur la touche ESC.

3.1.0.3 Définition des propriétés de chaque "port"

Pour définir les propriétés des *ports* ; soit son nom, son type ainsi que la taille, allez dans l'onglet *Interface* ou faire un double-clic sur un triangle représentant un port. Dans la fenêtre qui apparaît, entrez les valeurs afin d'obtenir le même résultat que celui de la fenêtre de la figure 3.4.

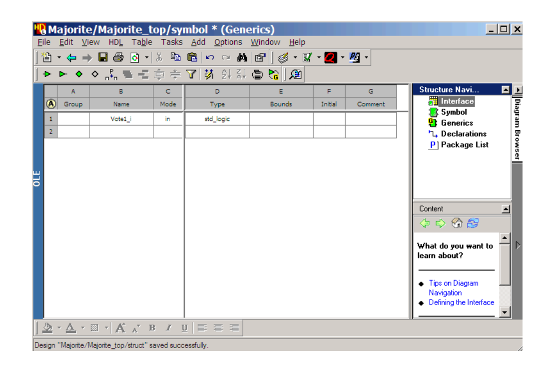


FIGURE 3.4 – Définition des propriétés des ports du composant

Les propriétés :

- Nom
  - Nom du signal. Selon la convention définie au ReDS le nom doit commencer par une majuscule. Il doit aussi avoir un suffixe définissant le "sens" du port, soit :
    - *_i pour les entrées
    - *_o pour les sorties
    - *_io pour les signaux bi-directionnelles
- Mode
  - Définition du "sens" du "port" (in, out ou inout). Le mode buffer n'est pas utilisé au ReDS
- Type
  - Permet de définir le type du port, soit Std_Logic ou Std_Logic_Vector (les autres types ne sont pas utilisés dans les entités/symboles)
- Bounds
  - Permet de définir la taille d'un vecteur (nombre de bits)
- Comment
  - Ajout un commantaire définissant le signal

3.2 Création d'une architecture

HDL Designer permet la création d'un certain nombre de types d'architectures différentes. Il permet aussi de créer/gérer plusieurs architectures pour un symbole donné.

3.2.1 Création d'une nouvelle architecture

Effectuez un double-clic sur le symbole créé au point précédent. Le dialogue de la figure 3.5 apparaît.

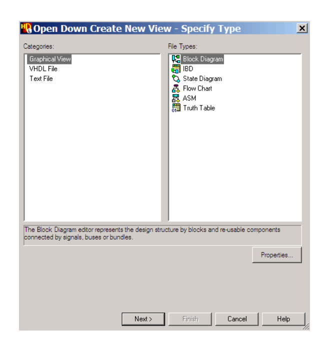


FIGURE 3.5 – Dialogue pour la création d'une nouvelle architecture

Si vous désirez créer une achitecture de type :

- Structurelle
  - Sélectionner "*Bloc Diagram*" et cliquez avec le bouton *Next>* Dans la nouvelle fenêtre, validez avec le bouton *Finish* (utilisation du nom par défaut) Suivre les instructions du § 3.4 "Création d'un vue structurelle"
- Textuelle en VHDL
  - Sélectionner "*Architecture*" et cliquez avec le bouton *Next>* Dans la nouvelle fenêtre, donnez un nom à cette architecture sous *Architecture* : Validez avec le bouton *Finish*, le fichier VHDL s'ouvre automatiquement
- Machine d'état graphique
  - Sélectionner "*State Diagram*" et cliquez avec le bouton *Next>* Dans la nouvelle fenêtre, validez avec le bouton *Finish* (utilisation du nom par défaut) Suivre les instructions du § 3.5 "Création de machine d'état"

Remarque : Les autres types d'architecture ne sont pas utilisés dans le cadre de l'institut ReDS.

3.2.2 Création d'une deuxième (ou plus) architecture

- Depuis le "*Design Explorer*", faite un clic-droite sur le composant auquel vous désirez ajouter une architecture
- Cliquez sur New puis choisissez l'architecture souhaitée en vous déplaçant dans le menu contextuel. Exemples :
  - *Architecture Structurelle*
    - *New* ▷ *Graphical View* ▷ *Bloc Diagram...*
  - *Textuelle*
    - *New* ▷ *VHDL View* ▷ *VHDL Combined...*
  - *Machine d'état*
    - *New* ▷ *Graphical View* ▷ *State Diagram...*

Dans la fenêtre qui apparaît, donnez un nom à la vue ainsi créée et validez avec le bouton *Finish* ou *OK.*

3.2.3 Gestion des architectures

Il est nécessaire d'indiquer l'architecture utilisée lorsqu'il y en a plusieurs. Un seule architecture est utilisée par entité. L'architecture utilisée (par défaut) pour une entité est illustrée par un triangle bleu à la gauche de celle-ci (cf. figure 3.6).

Pour changer l'architecture utilisée :

- Clic-droite sur l'architecture que vous voulez utiliser.
- Dans le menu contextuel, choisir l'option *"Set Default View"*.

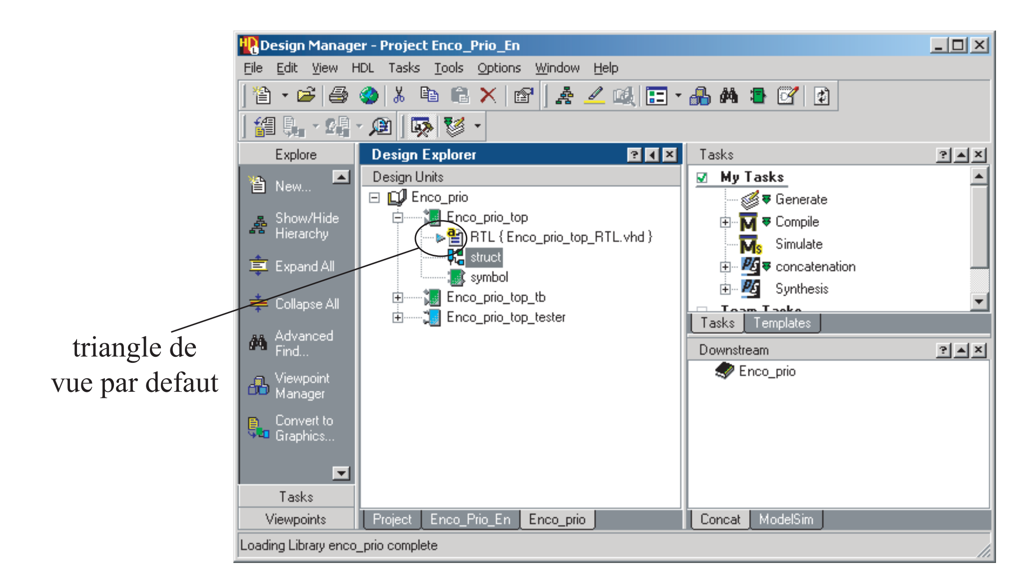


FIGURE 3.6 – Architecture/vue par defaut

3.3 Utilisation d'architectures multiples

Il est possible d'avoir plusieurs schémas (architectures) pour un même composant, exemple avec un multiplexeur 2 vers 1. Ce dernier a trois architectures (cf. figure 3.7) :

- Schéma avec des portes logiques de base.
- Schéma avec des portes NAND2.
- Description textuelle VHDL.

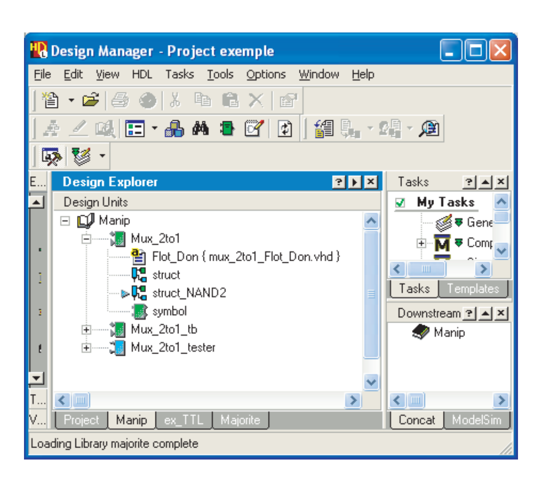


FIGURE 3.7 – Visualisation des architectures disponibles pour le composants Mux_2to1

Les figures suivantes montrent les schémas réalisés avec des portes logiques de base (figure 3.8) et réalisé avec des portes NAND2 (figure 3.9).

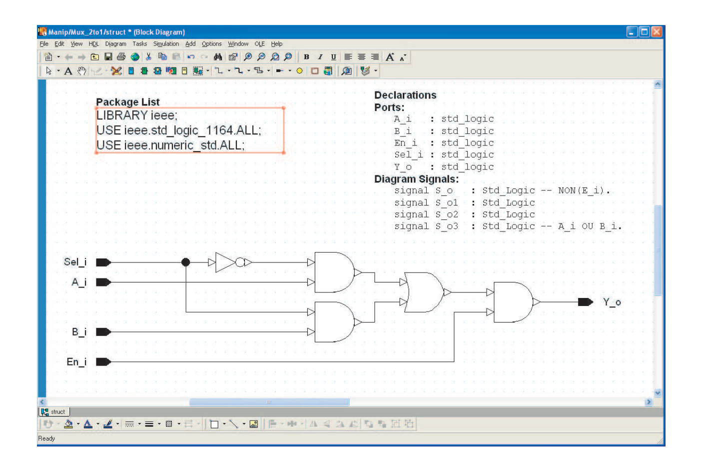


FIGURE 3.8 – Schéma réalisé avec des portes logiques de base

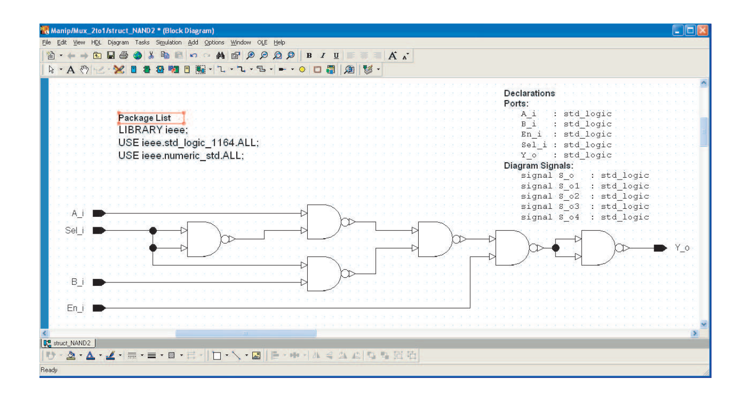


FIGURE 3.9 – Schéma réalisé avec des portes logiques de type NAND2

3.4 Création d'une vue structurelle

3.4.1 Outils indispensables

	Add Block	L'outil d'ajout de blocs
	Add Component	L'outil d'ajout de composants
	Add Embedded Block	L'outil d'ajout d'expressions concurrentes
	Add Signal	L'outil d'ajout de signaux à un élément
	Add Bus	L'outil d'ajout de vecteurs (signaux multi-éléments)
A	*Text*	L'outil d'ajout de commentaires

TABLE 3.2 – Outils indispensables

3.4.2 Ajout d'expressions concurrentes

Après avoir ajouté une expression concurrente, il faut effectuer un double-clic sur le bloc la représentant ; cette action aura comme effet d'afficher le dialogue suivant :

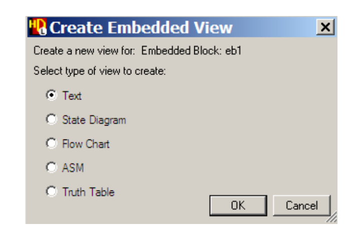


FIGURE 3.10 – Dialogue de création d'une expression concurrente

Choisir le type *"Text"*. Seules les instructions concurrentes de type *Text* sont intéressantes (et utilisées) dans le cadre du laboratoire de systèmes numériques !

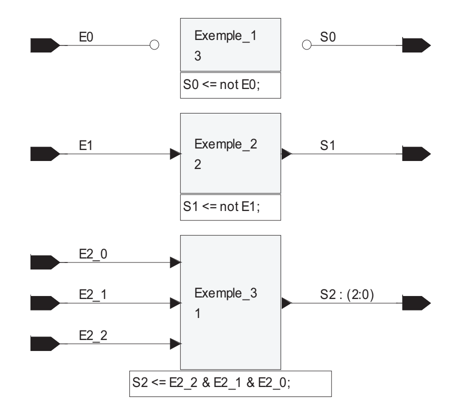


FIGURE 3.11 – Exemples d'instructions concurrentes

3.4.3 Différence entre les blocs et les composants

Il y a trois différences majeures entre un bloc et un composant :

- Le bloc permet d'avoir une approche de conception *"Top-Down"* (de haut en bas) tandis que le composant permet d'avoir une approche de conception *"Bottom-Up"* (de bas en haut).
- Une fois la décomposition finie, la conversion d'un bloc en un composant se fait facilement, l'inverse est impossible.
- Le bloc ne peut être instancié qu'à un seul et unique endroit. Par contre un composant peut être instancié un nombre quelconque de fois.

3.4.4 Remarque sur le terme "bus"

La société *MentorGraphics* a fait un abus de langage en utilisant le terme bus pour désigner un vecteur, soit un signal (au sens VHDL) composé de plusieurs éléments. Lorsqu'on lit le terme "*bus*" dans *"HDL Designer"*, il faut l'interpréter comme un vecteur.

Pour changer le nom des signaux internes, il suffit d'effectuer un double-clic sur la connexion représentant le signal.

Pour donner le nom au bloc, il suffit d'effectuer un double-clic sur le texte *"<block>"* et d'introduire le nom voulu à la place.

3.4.5 Style des connexions

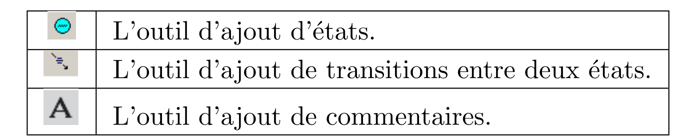
Il est possible de différencier, graphiquement, les vecteurs des signaux à un seul élément en fonction de la largeur du trait représentant la connexion.

Pour les signaux à un seul élément il faut utiliser le style *"Signal"* et pour les vecteurs de signal il faut utiliser le style "*Bus*" !

3.5 Création de machine d'état

HDL Designer permet de faire la création de machine d'état sous forme graphique.

3.5.1 Outils indispensables



	L'outil d'ajout d'états.
	L'outil d'ajout de transitions entre deux états.
A	L'outil d'ajout de commentaires.

Table 3.3 – Outils indispensables

3.5.2 Signification des différents symboles d'un graphe Etat

s0	Etat initial. La machine d'états est mise dans cet état lors d'un reset. Il ne peut y avoir qu'un état initial par graphe d'états.
s1	Un des états du graphe d'états Chaque état peut devenir l'état initial en cochant la casse "Start state" dans les propriétés de l'état.

Table 3.4 – Outils indispensables

3.5.2.1 Transition

Dans le menu contextuel, il est possible d'ajouter ou de supprimer des points de contrôle (définissant le tracé de la transition) à l'aide des commandes "*Add Route*" et "*Remove Route*". Il est également possible de changer le sens de la transition à l'aide de la commande *Reverse Direction*.

3.5.2.2 Condition de transition

Elle est contenue dans un rectangle en surimpression de la transition.

L'expression de la condition doit être écrite en VHDL.

Pour chaque état, il ne peut y avoir, partant de cet état, qu'une seule transition n'ayant aucune condition spécifiée. Cette transition sera effectuée par défaut lorsque aucune autre des conditions des transitions partant de cet état n'est satisfaite.

Si toutes les transitions partant d'un état ont une condition définie, une transition sans condition sera définie de manière implicite par le logiciel *HDL Designer* afin de maintenir l'état courant dans le cas où aucune des conditions ne serait satisfaite.

3.5.2.3 Numéro de priorité (pour les transitions)

Ce numéro est contenu dans un cercle en surimpression de la transition. Il devient visible dès qu'il y a plus d'une transition partant depuis un état.

Il indique l'ordre dans lequel l'évaluation des conditions de transition à partir d'un état doit être faite. Ce numéro permet de simplifier l'expression d'une condition de transition. Il sous-entend que cette condition n'est prise en compte que lorsque celles portant un numéro plus petit ne sont pas satisfaites. L'ordre d'évaluation des conditions peut être changé dans le dialogue des propriétés des transitions.

3.5.2.4 Action sur les sorties de type Moore

Pour définir une action sur les sorties de type *Moore*, il faut double-cliquer sur l'état auquel on veut associer l'action. Celle-ci doit être écrite en VHDL. Les actions sur les sorties sont affichées à côté des états. Usuellement, les actions sont déplacées afin d'être positionnées dans l'état auquel elles se rapportent. Action sur les sorties de type *Mealy* Pour définir une action sur les sorties de type *Mealy*, il faut double-cliquer sur la transition à laquelle on veut associer l'action. Celle-ci doit être écrite en VHDL. Les actions sur les sorties sont affichées dans le même rectangle que la condition de transition, dont elles sont séparées par une ligne (les conditions apparaissent au dessus et les actions au-dessous de cette ligne).

Les actions sur les sorties de type Mealy peuvent également être spécifiées dans les états comme pour les sorties de type Moore. Ces actions contiennent, dans ce cas, une affectation conditionnelle permettant de définir la valeur des sorties selon une combinaison des entrées.

3.5.3 Propriétés essentielles d'un graphe d'états

Les propriétés d'un graphe d'états se trouvent dans les menus *"Edit ▷ Object Properties..."* et *"Diagram ▷ State Machine Properties. . . "*.

3.5.3.1 La fenêtre "Object Properties"

Cettre fenêtre vous permet de définir les paramètres essentiels de la machine d'état tel :

- le signal à utiliser pour l'horloge ainsi que le flanc de déclenchement.
- le signal à utiliser pour le reset, son niveau d'activation ainsi que son mode (reset synchrone ou asynchrone).

3.5.4 Autres propriétés d'un graphe d'états

Les propriétés d'un graphe d'états se trouvent dans les menus *"Diagram ▷ State Machine Properties. . . "*.

3.5.4.1 Generation

Dans cet onglet on peut spécifier :

- si l'on veut une machine d'états synchrone ou asynchrone. Dans le cadre des laboratoires, on utilise uniquement des machines d'états synchrones.
- le nom des signaux internes représentent l'état présent et l'état futur. Si les deux champs correspondants sont vides, des noms par défaut (*current_state* et *next_state*) sont utilisés. L'intérêt, de spécifier des noms à ces signaux, est de pouvoir effectuer des affectations sur les sorties à partir de bits d'états.
- . . .

3.5.4.2 Encoding

Permet de spécifier la manière dont il faut coder les états. En choisissant l'option :

- *Auto*, les états seront codés selon les options sélectionnées par l'utilisateur dans le synthétiseur.
- *Specified*, l'utilisateur peut spécifier le codage de chaque état dans "*HDL Designer*" en allant dans les propriétés des états.
- les autres choix ne sont pas très intéressants dans le cadre du laboratoire.

3.5.4.3 Statement Blocks

Les *Global Actions* permettent de spécifier des actions placées au début du processus de sorties (quel que soit l'état courant).

Les *Concurrent Statements* permettent de définir des instructions concurrentes au graphe d'états.

Les *State Register Statements* permettent de spécifier les actions à effectuer sur le flanc (montant ou descendant) de l'horloge. Si une action est spécifiée dans ce champ, il faut également effectuer explicitement l'affectation de l'état futur à l'état présent. Si ce champ est laissé vide, l'affectation de l'état futur à l'état présent est effectuée de manière implicite.

Affectation d'une sortie à un bit d'état Il est parfois nécessaire d'avoir des sorties qui n'ont pas d'aléas (par exemple lorsque elles commandent des bascules RS). Un des moyens est d'avoir les sorties qui correspondent directement à un bit d'état. Pour ce faire, il faut :

- Coder les états de façon qu'un bit correspond toujours à la valeur voulue pour une sortie (cf. § "Encoding")
- Dans *"Concurrent Statements"* de l'onglet "*Statement Block*" des propriétés de la machine ajoutez :

$$maSortie \Leftarrow Etat_Present(x);$$

  - ma sortie : nom de la sortie en question.
  - Etat_Present : c'est le nom donner à l'état en court. Il est spécifier dans l'onglet "*Generation*" des propriétés de la machine.

3.5.4.4 Declarations Blocks

Les déclarations faites sous cet onglet apparaîtront dans la partie déclarative de l'architecture. C'est dans celle-ci que l'on trouve, entre autres, la déclaration des signaux internes comme, par exemple, l'expression ci-dessous :

signal Capt : Std_Logic_Vector(1 downto 0);

Process Declarations Les déclarations faites sous cet onglet apparaîtront dans la partie déclarative du processus de mémorisation ainsi que du processus de sorties.

3.5.4.5 Signals Status

Permet de spécifier les caractéristiques des signaux internes et des sorties. Les valeurs spécifiées dans le champ "*Default value*" sont affectées aux signaux correspondants au début du processus de sorties.

Remarque : Cette assignation de valeurs par défaut permet d'éviter la création de "latches" par inadvertance, lorsque l'on oublie de spécifier un des cas dans l'affectation des sorties ou si le nombre d'état de la machine est inférieur au nombre maximum possible (soit 2n où n est le nombre de bit de la machine).

Le champ *"Status"* permet de choisir entre trois possibilités :

– "*COMBINATORIAL*" : La sortie est calculée de façon combinatoire à partir des bits d'état et, pour une sortie de type Mealy, d'une combinaison des entrées. C'est, généralement, cette possibilité que l'on utilise pour la réalisation de graphes d'états.
– *"REGISTERED"* : La sortie est calculée dans le décodeur de sortie puis elle est synchronisée par un registre ce qui permet d'obtenir une sortie sans transitoire. Cette possibilité a comme inconvénient de rajouter une bascule supplémentaire par sortie.
– *"CLOCKED"* Le comportement des sorties obtenu est exactement le même qu'avec le status *"REGISTERED"* et il y a également adjonction d'un flip-flop par sortie. Cependant alors que dans l'option "*REGISTERED*" un signal interne correspondant à la sortie avant synchronisation est disponible, ce n'est pas le cas dans l'option "*CLOCKED*". Le signal interne est directement celui issu des flip-flops de post-synchronisation, donc identique à celui de sortie. Avec cette option, il n'est pas nécessaire de spécifier une valeur de sortie par défaut.

3.5.5 Exemple d'une machine d'état

Nous allons illustrer les points précédents en prenant comme exemple un détecteur de flancs.

Ce système est composé d'une entrée *Signal* et de deux sorties *Pulse* et *S_Active*, ainsi qu'une entrée pour l'horloge (*Clk*) et une entrée pour le reset (*Reset*).

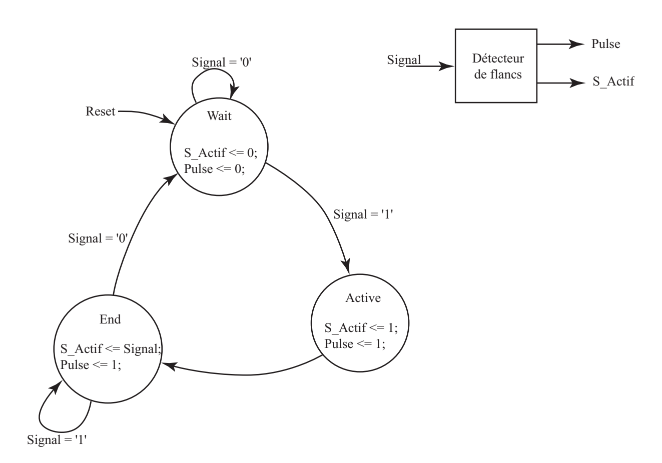


FIGURE 3.12 – Schéma de conception de cette machine d'état

La sortie *Pulse* fournit une pulse lors du passage de l'entrée *Signal* de *'0'* à *'1'*. Cette pulse à une durée d'une période d'horloge.

La sortie *S_active* a le même comportement que *Pulse*, elle s'active au même moment, mais se désactive lorsque Signal passe de *'1'* à *'0'*. La durée minimum est d'une période d'horloge.

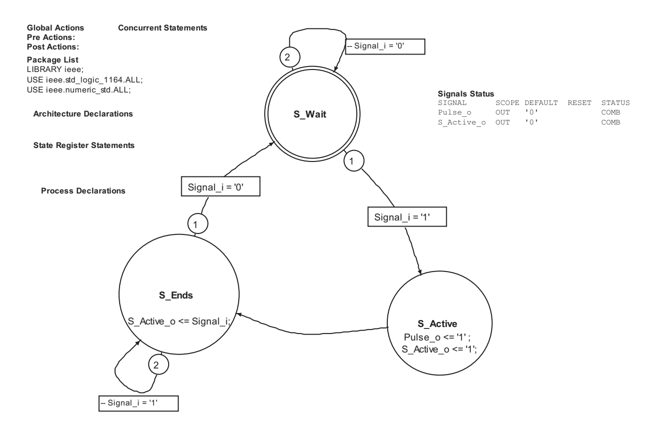


FIGURE 3.13 – Machine du détecteur de flancs réalisée sous *HDL Designer*

Le code VHDL généré par HDL Designer résultant de la figure 3- 12 est donnée à la figure figure 3-13.

```
architecture fsm of Detecteur_Flancs is

   -- Architecture Declarations
   type state_type is (
      S_Wait,
      S_Active,
      S_Ends
   );

   -- State vector declaration
   attribute state_vector : string;
   attribute state_vector of fsm : architecture is "current_state" ;


   -- Declare current and next state signals
   signal current_state : state_type ;
   signal next_state : state_type ;

begin

   ----------------------------------------------------------------------
   clocked : process(
      Clk_i,
      Reset_i
   )
   ----------------------------------------------------------------------
   begin
      if (Reset_i = '1') then
         current_state <= S_Wait;
         -- Reset Values
      elsif (Clk_i'event and Clk_i = '1') then
         current_state <= next_state;
         -- Default Assignment To Internals

      end if;

   end process clocked;

   ----------------------------------------------------------------------
   nextstate : process (
      Signal_i,
      current_state
   )
   ----------------------------------------------------------------------
   begin
      case current_state is
      when S_Wait =>
         if ( Signal_i = '1' ) then
            next_state <= S_Active;
         else
            next_state <= S_Wait;
         end if;
      when S_Active =>
            next_state <= S_Ends;
      when S_Ends =>
         if ( Signal_i = '0' ) then
            next_state <= S_Wait;
         else
            next_state <= S_Ends;
         end if;
      when others =>
         next_state <= S_Wait;
      end case;

   end process nextstate;

   ----------------------------------------------------------------------
   output : process (
      Signal_i,
      current_state
   )
   ----------------------------------------------------------------------
   begin
      -- Default Assignment
      Pulse_o <= '0';
      S_Active_o <= '0';
      -- Default Assignment To Internals

      -- Combined Actions
      case current_state is
      when S_Active =>
          Pulse_o <= '1' ;
         S_Active_o <= '1';
      when S_Ends =>
          S_Active_o <= Signal_i;
```

```
      when others =>
         null;
      end case;

   end process output;

   -- Concurrent Statements

end fsm;
```

Listing 3.1 – Code généré par HDL Designer

3.6 Cohérence des symboles (composants ou blocs)

L'interface entre les différentes vues d'un symbole et les différents emplacements où celui-ci peut-être instancié, est définie par les *ports* se trouvant sur le composant ou par les connexions arrivant sur le bloc.

Lorsque cette interface est modifiée, la cohérence entre ses différentes vues et ses instanciations n'est pas toujours assurée. Dans certains cas, il s'avère nécessaire de rétablir la cohérence de manière explicite.

Les points suivants expliquent la manière de procéder afin de maintenir la cohérence lors de la modification d'une interface.

3.6.1 Cohérence entre un symbole et ses vues

3.6.1.1 Schéma bloc

Pour rétablir la cohérence entre un symbole et son schéma bloc, il faut aller dans le schéma bloc, s'assurer qu'aucun symbole (composant ou bloc) n'est sélectionné, puis aller dans les menus *"Diagram"* ▷ *"Update"* ▷ *"Interface..."*. Le dialogue suivant apparaît :

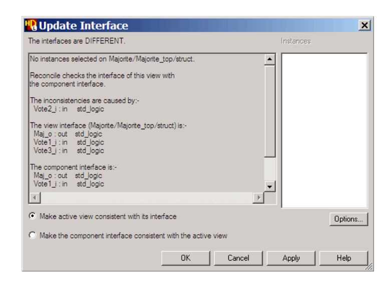


Figure 3.14 – Fenêtre pour la cohérence entre un symbole et une de ces vues

Si le schéma-bloc n'est pas cohérent avec son symbole, le texte "*The Interfaces are DIFFERENT*" apparaît au sommet du dialogue. Dans le cas contraire, le texte "*Interfaces are IDENTICAL*" apparaît au sommet de celui-ci.

Lorsque le schéma-bloc n'est pas cohérent avec son symbole, il faut sélectionner le choix "*Make active view consistent with its interface*" puis cliquer sur le bouton "*OK*". Cette opération va modifier les "ports" du schéma bloc afin qu'ils correspondent à ceux présent dans le symbole (composant ou bloc).

3.6.1.2 Description VHDL

Pour maintenir la cohérence entre une description VHDL et son symbole, il faut s'assurer que le nom spécifié dans la déclaration de l'architecture correspond bien au nom du symbole (composant ou bloc).

```
architecture Comport of Nom_Symbole is
begin
end Comport;
```

Listing 3.2 – Description VHDL

3.7 Importation d'une description textuelle VHDL

Il est possible d'importer des fichiers contenant une description VHDL (une entité et une architecture), dans une des bibliothèques associées à un projet.

Lors de l'importation (ou convertion), on peut demander que ces descriptions soient représentées, à condition que cela soit possible, sous forme de schéma-bloc, de graphe d'états ou de description VHDL (textuelle).

1. Pour importer des descriptions VHDL dans une bibliothèque, il faut aller dans la bare de gauche *Main* ▷ *New/Add*. Suite à cette opération, la fenêtre de la figure 3.15 s'ouvre.
2. Sélectionnez les options <*Add Existing Files to Library*>.
3. Sélectionnez la librairie qui va recevoir le fichier vhdl.
4. Cliquez sur *Next* >. Suite à cette opération, la fenêtre de la fiugre 3.15 s'ouvre.

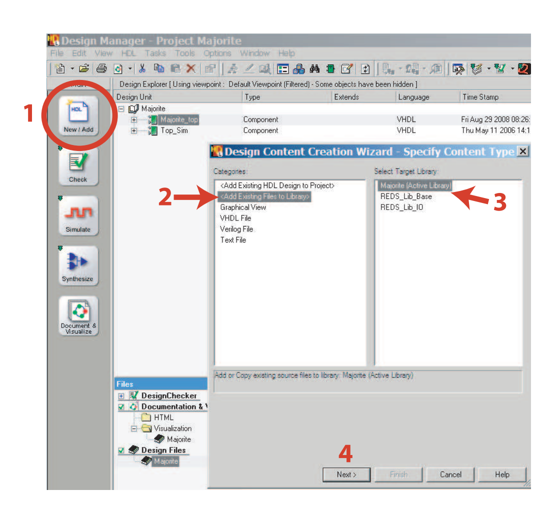


Figure 3.15 – Ajout de sources VHDL

Une fois la fenêtre "*Add Existing Files To Library...*"(figure 3.16) ouverte :

1. Sélectionnez "*Copy specified files*" sous la rubrique "*Addition Method*". Ceci aura pour but de copier le fichier VHDL dans le répertoir *VHDL* de votre librairie.
2. Vérifiez que le répertoire de destination et bien le dossier VHDL de votre librairie.
3. Rechercher le répertoire contenant le fichier VHDL original au moyen de la fenêtre *Folders* ou du bouton *Browse*.
4. Sélectionnez le ou les fichiers VHDL à copier en cochant les cases correspondantes.
5. Cliquez sur *OK*.

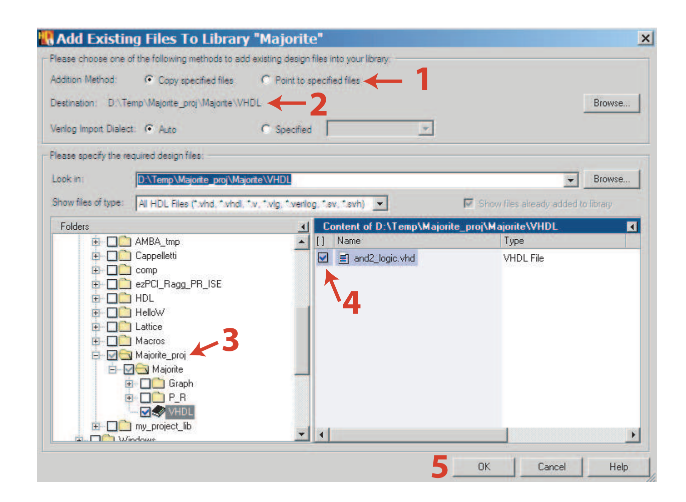


FIGURE 3.16 – Sélection fichiers vhdl à ajouter dans le projet

4 Simulation avec ModelSim

Maintenant que notre schéma est terminé, nous pouvons le vérifier. Le simulateur, que nous utilisons travaille avec des descriptions VHDL. La première étape consistera à générer la description VHDL à partir du schéma et à le compiler afin de pouvoir les simuler avec *ModelSim*.

Il y a deux méthodes pour simuler le circuit précedemment décrit :

- *La simulation manuelle*
  - Elle consiste à donner des valeurs de stimulation aux entrées de manière manuelle et de vérifier visuellement les résultats obtenus. L'assignation des stimulis et la visualisation des signaux se fait dans *HDL Designer* et à l'aide d'une console graphique.
- *La simulation automatique*
  - Celle-ci utilise un banc de test. Ce banc de test génére des stimulations pour le circuit à tester et génère des valeurs de référence. Ces valeurs de référence sont comparées aux valeurs fournies par le composant testé (simulé). Si les valeurs obtenues et les références ne sont pas identiques, des messages d'erreur sont générés automatiquement.

4.1 Simulation manuelle

4.1.1 Présentation de la console de simulation

Pour réaliser une simulation manuelle, il faut pouvoir assigner des valeurs aux entrées et visualiser les valeurs obtenues sur les sorties. Ces tâches sont réalisées à l'aide de la console *REDS_console* (cf. figure 4.1).

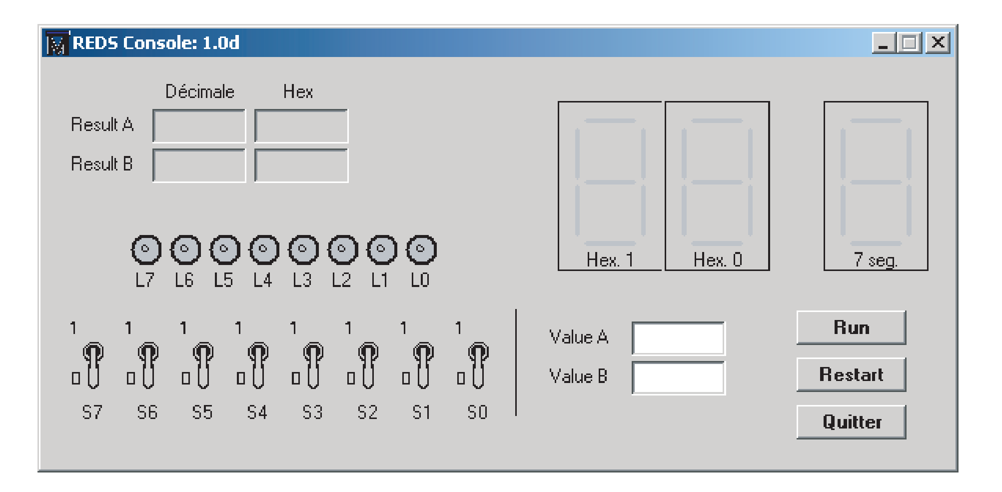


FIGURE 4.1 – Console pour la simulation manuelle

Les liens entre les signaux disponibles dans le composant à tester et les signaux de la console sont

réalisés dans le composant *Top_Sim* (disponible dans la bibliothèque de travail et la bibliothèque *REDS_Lib_IO*). Ce composant contient tous les signaux disponibles de la console (cf. figure 4.2).

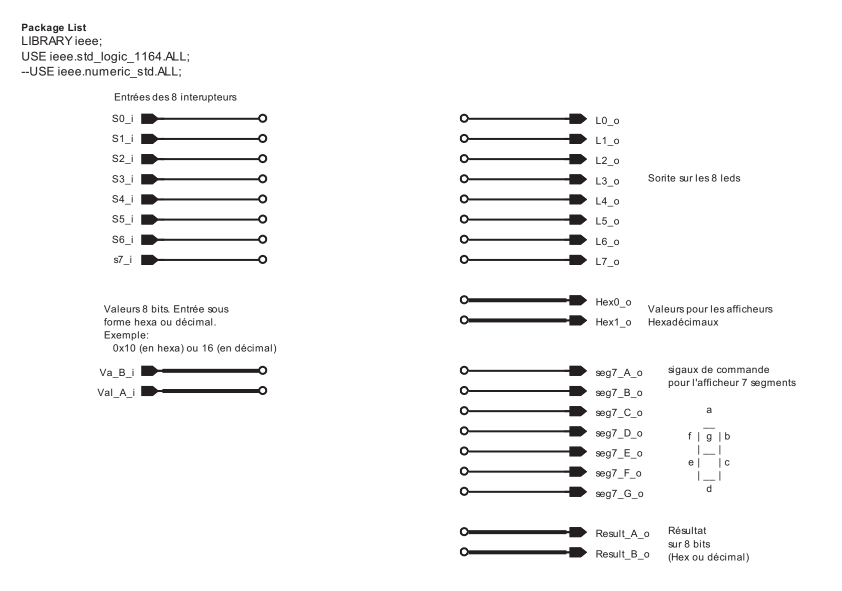


FIGURE 4.2 – Signaux diponible dans le composant Top_Sim

4.1.2 Signaux disponibles

Cette console est composée des éléments suivants :

- Une série de 8 interrupteurs, *S0* à *S7* liés aux signaux *S0_i* à *S7_i* du composant *Top_Sim.*
- Deux zones pour l'affectation d'une valeur sur 8 bits, *Val A* et *Val B*. Il est possible d'entrer les valeurs sous forme décimale (de 0 à 255) ou sous forme hexadécimale (0x00 à 0xFF). Les signaux associés sont les vecteurs *Val_A_i* et *Val_B_i*, tout deux de 8 bits. ***Remarque :*** si la valeur d'entrée est erronée, la console force les entrées à la valeur non initialisée ("UUUUUUUU").
- Une série de 8 led, *L0* à *L7*, liées aux signaux *L0_o* à *L7_o.*
- Deux zones de lecture de valeurs 8 bits *(Result A* et *Result B*). Ces valeurs sont affichées sous la forme hexadécimale et décimale. Les signaux associés sont les vecteurs *Result_A_o* et *Result_B_o*, tout deux de 8 bits. ***Remarque*** : Si la valeur fournie par le composant n'est pas une valeur comprise entre 0 et 255 (par exemple une valeur non initialisée, "UUUUUUUU"), rien n'est affiché.
- Deux afficheurs hexadécimaux, liés aux signaux *Hex0_o* et *Hex1_o* (signaux de 4 bits). Ils permettent d'afficher les valeurs de 0 à 9 et de A à F.
- Un afficheur 7 segments. Chaque segment de cet afficheur est lié avec un signal indépendant (de *Seg7_a_o* à *Seg7_g_o*). Ceci permet de commander chaque segment individuellement selon la figure 4.3.

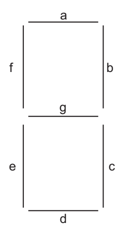


FIGURE 4.3 – Décomposition d'un afficheur 7 segments

- Les boutons de commande
  - *Run* : Validation des affectations réalisées sur les entrées, avance le temps simulé de 100 ns (*ModelSim* évalue l'état des sorties) puis affichage des résultats obtenus.
  - *Restart* : Relance une simulation. Lors du relancement d'une simulation, les derniers fichiers modifiés, par exemple suite à la correction d'une erreur, sont rechargés.

– *Quitter* : Quitte le simulateur.

4.1.3 Connexion de la console

Pour préparer une simulation manuelle, il faut connecter le composant à simuler dans *Top_Sim*. Pour ce faire, veuillez suivre les points suivants :

– Sélectionnez la bibliothèque de travail à l'aide des onglets.
– Double-cliquez sur le composant *Top_Sim*.
– Dans la nouvelle fenêtre, ajoutez (instanciez) le composant que vous voulez tester en ouvrant le *Component Browser* en utilisant l'outil d'ajout de composants (, *Add component*).
– Faites les liaisons entre les entrées/sorties de votre composant et les signaux de *Top_Sim*.
– Sauvegardez les modifications.

La figure 4.4 donne une vue du résultat obtenu pour le composant *Majorite_top*.

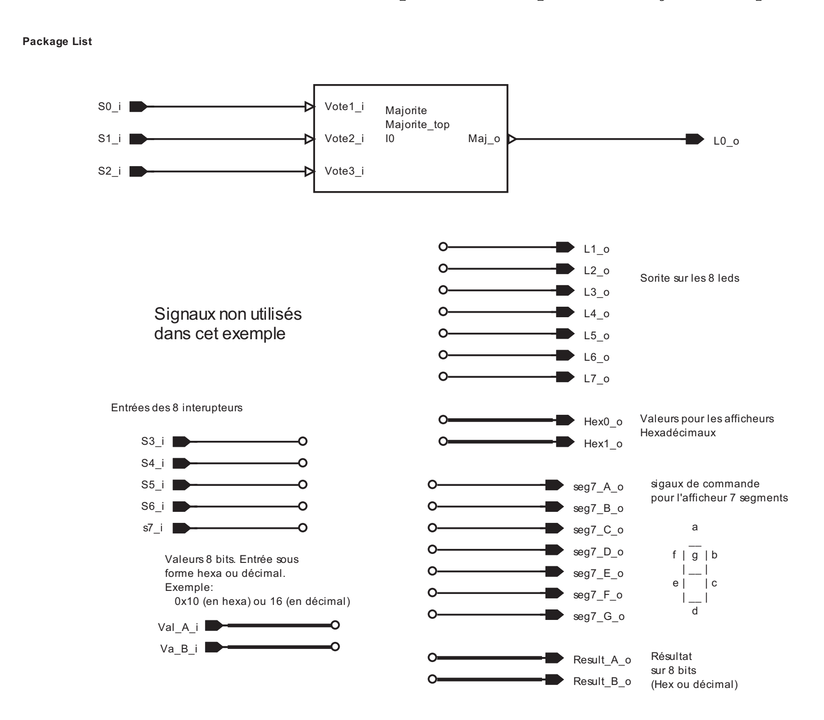


FIGURE 4.4 – Composant *Top_Sim* intégrant le composant *Majorite_top*

Dans le cas où le composant *Top_Sim* n'éxiste pas dans votre projet, il est possible de le copier depuis la bibliothèque *REDS_Lib_IO*. La marche à suivre est la suivante :

– ouvrir la librairie *REDS_Lib_IO*.
– sélectionner le composant *Top_Sim* et le copier (menu *Edit* ▷ *Copy*).
– aller dans bibliothèque *Nom_Projet* (important : il faut sélectionner le *top* de la bibliothèque).
– coller le composant *Top_Sim* (menu *Edit* ▷ *Paste*).

4.1.4 Génération, compilation

Les étapes de génération et de compilation permettront de vérifier que votre schéma est SYNTAXIQUEMENT correct. Cela ne veut en aucun cas dire que votre conception fonctionne !

Pour lancer la génération et la compilation, il faut :

– Retourner dans *Design Manager* et choisir l'onglet de votre librairie, dans notre exemple *Majorite*.
– Sélectionner le composant *Top_Sim*.

- Double-cliquez sur *"Generate "*dans la fenêtre *Tasks* (en haut à droite), cf.figure 4.5).
- Double-cliquez sur *"DesignChecker Flow "*dans la fenêtre *Tasks* (en haut à droite) ou cliquez sur "*Check*"(icone à gauche) , cf.figure 4.5).

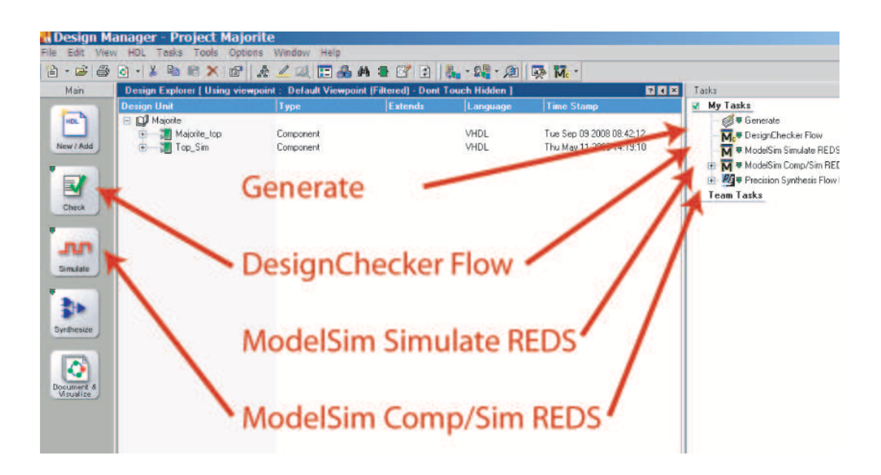


FIGURE 4.5 – Fenêtre des tasks

Cette étape provoquera l'ouverture d'une fenêtre de "*log*". Regardez s'il n'y a pas d'erreurs ou de warnings (lignes rouges) et que la dernière inscription porte la mention :

"Data preparation step completed, check transcript..."

Dans ce cas, votre description a une syntaxe correcte. Dans le cas contraire, il y a une/des erreur(s). Il faut donc corriger cette/ces erreur(s) et relancer la compilation !

4.1.5 Lancement du simulateur

Toujours en sélectionnant le composant *Top_Sim*, double-cliquez sur *"ModelSim Simulate REDS"* dans la fenêtre *Tasks* pour lancer le simulateur.

***Remarque* : cette action ne recompile pas le composant.**

Pour effectuer le flow compet (Generate, compile, run Modelsim), il faut double-cliquez sur *"ModelSim Comp/Sim REDS"* dans la fenêtre *Tasks*.

La fenêtre de dialogue qui apparaît permet de configurer certains paramètres comme le pas de simulation (délai entre chaque calcul des signaux). Les paramètres par défaut conviennent ▷Cliquer sur "*OK*".

ModelSim se lance. Attendez que les fenêtres soient complétement ouvertes.

Une fois le simulateur *ModelSim* lancé, ouvrez la console à l'aide du bouton *REDS_console* qui se trouve dans la fenêtre principale de *ModelSim* (cf. figure 4.6).

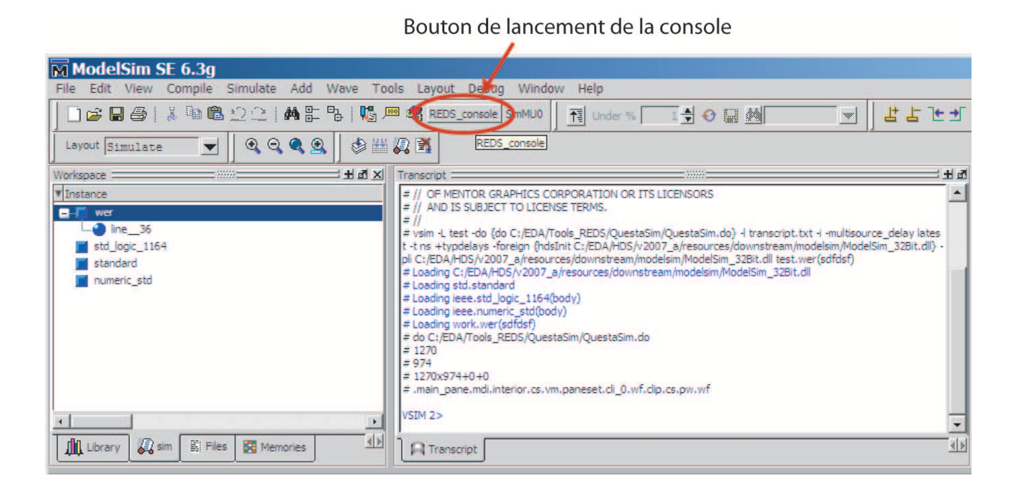


FIGURE 4.6 – Bouton de lancement de la console

Maintenant, vous devez retourner dans la fenêtre du schéma réalisé sous *HDL Designer* (pour notre exemple *Majorite_top*). Si nécessaire, vous devez rouvrir le schéma.

La barre d'outils utilisée pour la simulation manuelle apparaît au bas de cette fenêtre (cf. figure 4.7). Si ce n'est pas le cas, cliquez avec le bouton de droite dans l'espace libre laissé à côté d'une autre barre d'outils pour ouvrir le menu contextuel, et sélectionnez *Simulation Tools*.



FIGURE 4.7 – Barre d'outils pour la simulation

4.1.6 Les sondes

Nous allons utiliser des sondes pour visualiser les valeurs des signaux disponibles dans le schéma. Pour ajouter une sonde à un signal, il faut le sélectionner. Si vous avez plusieurs sondes à rajouter, vous pouvez faire une multi-sélection. Cliquez ensuite sur le bouton de la barre des outils de simulation.

4.1.7 Simulation et assignation des entrées

L'assignation des valeurs aux entrées se fait avec les boutons disponibles sur la console. Pour changer la valeur d'un bouton, il faut cliquer sur celui-ci. La visualisation des sorties se fait au travers des leds ou des afficheurs 7 segments ou hexadécimaux. Afin de valider les nouvelles valeurs choisies et d'obtenir l'état des sorties, il faut cliquer sur le bouton *Run*.

Astuce : Mettez la console devant la fenêtre du schéma, ainsi les deux fenêtres sont visibles et il est aisé de voir l'évolution des signaux selon les valeurs fournies aux entreés.

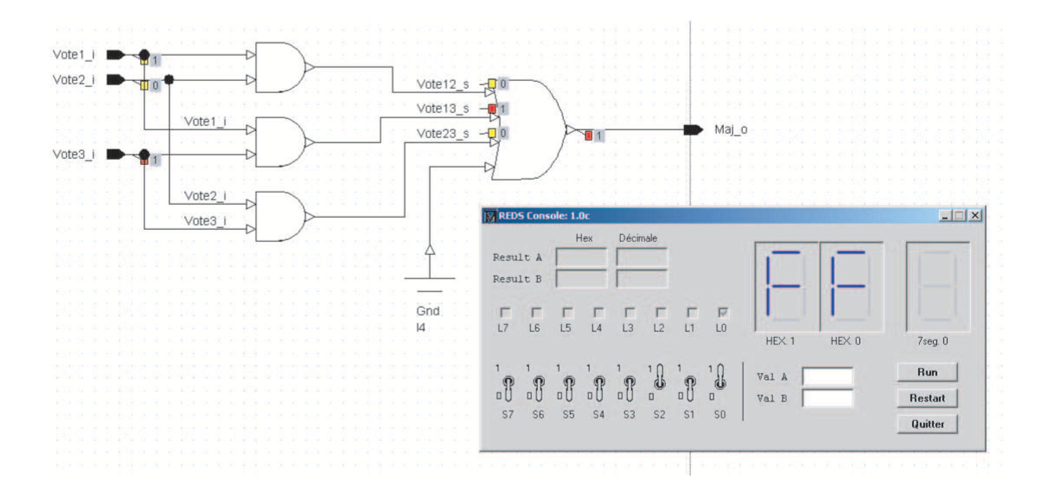


FIGURE 4.8 – Visualisation du schéma après l'ajout des sondes

Remarque : Les connexions entre les signaux de la console et du composant à tester sont réalisées dans le composant *Top_Sim*.

4.1.8 Preuve de la simulation manuel

Il est possible de voir l'évolution des signaux sur un chronogramme (cf. figure 4.9). Ceci permet de fournir une preuve des cas simulés manuellement. Dans le chronogramme il est nécessaire d'ajouter toutes les entrées et toutes les sorties du système en cours de vérification. Il est important de prendre les signaux du composant et pas ceux de *Top_Sim* afin de voir les noms des signaux dans le chronogramme. Vous pouvez vous référer aux explications du § 4.2.3, "Préparation en vue de la simulation" pour préparer le chronogramme.

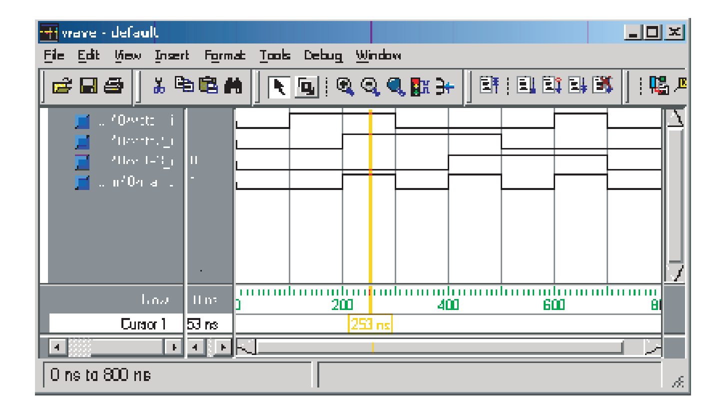


FIGURE 4.9 – Chronogramme de quelques pas lors d'une simulation manuel

Il faut aussi fournir la fenêtre *log* de *ModelSim* afin de connaître quels sont les composants qui ont été chargés.

```
# Loading C:/EDA/HDS/v2004_1/resources/downstream/modelsim/ModelSim_32Bit.dll
# Loading C:\EDA\ModelSim\v5_8c\win32/../std.standard
# Loading C:\EDA\ModelSim\v5_8c\win32/../ieee.std_logic_1164(body)
# Loading C:\EDA\ModelSim\v5_8c\win32/../ieee.numeric_std(body)
# Loading D:\Projects\OutilsEDA\Majorite_proj/Majorite/Comp.top_sim(struct)
# Loading D:\Projects\OutilsEDA\Majorite_proj/Majorite/Comp.majorite_top(struct)
# Loading D:\Projects\OutilsEDA\Majorite_proj/REDS_Lib_Base/Comp.and2(logic)
# Loading D:\Projects\OutilsEDA\Majorite_proj/REDS_Lib_Base/Comp.gnd(logic)
# Loading D:\Projects\OutilsEDA\Majorite_proj/REDS_Lib_Base/Comp.or4(logic)
```

Listing 4.1 – Log lors du chargement de Top_Sim avec le projet Majorite

4.1.9 Corrections

Si vous avez une correction à apporter au schéma, vous êtes obligés d'enlever les sondes. Pour cela, cliquez sur le bouton

Une fois votre schéma corrigé, relancez le script de compilation pour le simulateur (bouton *Compile* dans la fenêtre *Tasks*). Il n'est pas nécessaire de relancer le simulateur. Par contre, une remise à zéro est nécessaire, cliquez sur le bouton *Restart* de la console. Réassignez vos entrées et relancez la simulation.

4.1.10 Quitter le simulateur

Pour fermer le simulateur, cliquez avec le bouton *Quitter* de la console.

4.2 Simulation automatique

4.2.1 Génération et Compilation

Les étapes de génération et de compilation permettront de vérifier que votre schéma est SYNTAXIQUEMENT correct. Cela ne veut en aucun cas dire que votre conception fonctionne !

Pour lancer la génération et la compilation, il faut :

- Retrouner dans *Design Manager* et choisir l'onglet de votre librairie, dans notre exemple *Majorite*.
- Sélectionner le composant **_top_tb*. Dans le cas de l'exemple : *Majorite_top_tb*.
- Double-cliquez sur *"Generate* "dans la fenêtre *Tasks* (en haut à droite), cf.figure 4.10).
- Double-cliquez sur *"DesignChecker Flow* "dans la fenêtre *Tasks* (en haut à droite) ou cliquez sur "*Check*"(icone à gauche) , cf.figure 4.10).

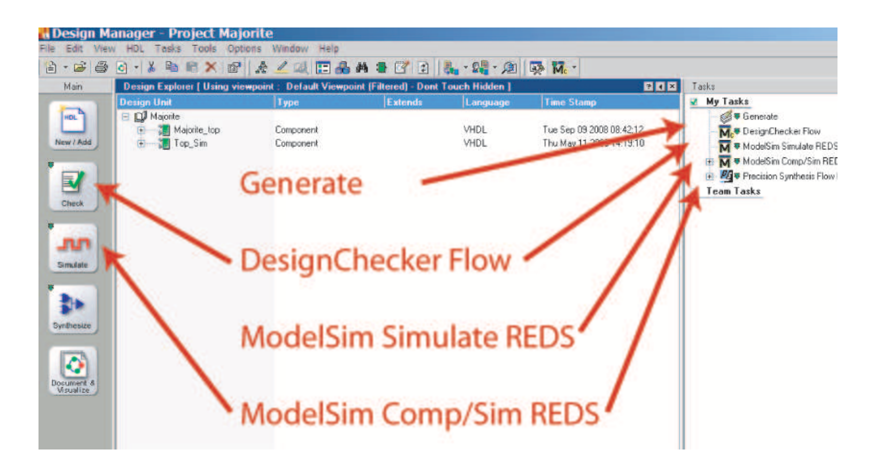


Figure 4.10 – Fenêtre des tasks

Cette étape provoquera l'ouverture d'une fenêtre de log. Regardez s'il n'y a pas d'erreurs ou de warnings (lignes rouges) et que la dernière inscription porte la mention :

"Data preparation step completed, check transcript..."

Dans ce cas, votre description a une syntaxe correcte. Dans le cas contraire, il y a une/des erreur(s). Il faut donc corriger cette/ces erreur(s) et essayer de compiler à nouveau !

4.2.2 Lancement du simulateur

Toujours en sélectionnant le composant **_tb* (dans le cas de l'exemple *Majorite_top_tb*), cliquez sur *"ModelSim Simulate REDS"* dans le fenêtre *Tasks*.

La fenêtre de dialogue qui apparaît permet de configurer certains paramètres comme le pas de simulation (délai entre chaque calcul des signaux). Les paramètres par défaut conviennent ▷ Cliquer sur *OK*.

ModelSim se lance. Attendez que les fenêtres soient complétement ouvertes.

4.2.3 Préparation en vue de la simulation

4.2.3.1 Ajout des traces dans le chronogramme

Sélectionnez les signaux dont on désire garder la trace, dans la fenêtre *signal*s puis déplacer-les, à l'aide de la souris, dans la fenêtre *wave*.

Il est possible de se déplacer dans la structure du composant simulé en sélectionnant les différents modules dans la fenêtre *structure* (situé dans la fenêtre principale). La fenêtre *signals* montre les signaux du module actuellement sélectionné, ce qui permet de les ajouter dans la fenêtre *wave*.

4.2.3.2 Ajout d'intercalaire (en anglais "divider")

Il est possible d'ajouter des intercalaires entre les différentes traces afin d'améliorer la lisibilité du chronogramme.

Pour ajouter un intercalaire, allez dans les menus *Insert* ▷ *Divider*... de la fenêtre wave ou effectuez un clic droite de la souris et sélectionnez le menu *Insert divider*. Le dialogue suivant apparaît :

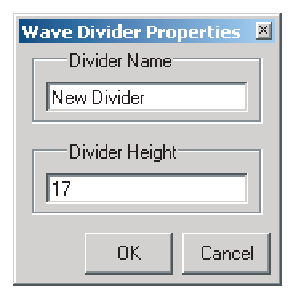


FIGURE 4.11 – Création d'un intercalaire

Dans ce dialogue, remplacer le texte *New Divider* par une chaîne de caractères caractérisant l'intercalaire. Cette chaîne est indicative et peut donc contenir n'importe quels caractères imprimables.

4.2.3.3 Formatage des traces

Pour effectuer la mise en forme d'une trace dans le chronogramme, il faut cliquer sur le bouton droite de la souris et sélectionner le menu "*Properties...*" sur le nom du signal que l'on veut changer le formatage. Le dialogue suivant apparaît :

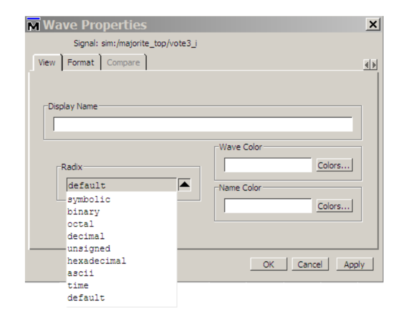


FIGURE 4.12 – Fenêtre de formatage des traces d'un chronogramme

Les champs intéressants, pour les systèmes logiques, permettent de spécifier :

- *Radix*
  - la base de la représentation du signal. Ce qui permet, par exemple, de représenter la valeur entière d'un compteur à l'aide du choix "*Unsigned*".
- *Display Name*
  - permet de spécifier le nom à afficher à la place du nom du signal.

4.2.4 Lancement de la simulation

Aller dans les menus *Simulate* ▷ *Run* ▷ *Run -all* de la fenêtre principale ou cliquer sur le bouton se trouvant dans la fenêtre *wave* afin d'effectuer la simulation.

La simulation s'arrêtera dès que les valeurs des entrées et des sorties resteront constantes, typiquement lorsqu'un "*wait ;*" est rencontré dans la description VHDL du test bench.

4.2.5 Preuve de la simulation

La preuve d'une simulation est donnée dans la fenêtre *Log* de *Model*Sim (fenêtre principale de l'outil). C'est dans cette zone que *ModelSim* informe l'utilsateur. Les informations importantes sont :

- La liste des fichiers chargés et donc simulés.
- Les messages fournis par le test-bench, par exemple le nombre d'erreur détectées.

```
# Loading C:/EDA/HDS/resources/downstream/modelsim/ModelSim_32Bit.dll
# Loading C:\EDA\ModelSim\win32/../std.standard
# Loading C:\EDA\ModelSim\win32/../ieee.std_logic_1164(body)
# Loading C:\EDA\ModelSim\win32/../ieee.numeric_std(body)
# Loading D:\Projet\Majorite_proj/Majorite/Comp.majorite_top_tb(struct)
# Loading D:\Projet\Majorite_proj/Majorite/Comp.majorite_top(struct)
# Loading D:\Projet\Majorite_proj/ReDS_lib_Base/Comp.and2(logic)
# Loading D:\Projet\Majorite_proj/ReDS_lib_Base/Comp.gnd(logic)
# Loading D:\Projet\Majorite_proj/ReDS_lib_Base/Comp.or4(logic)
# Loading D:\Projet\Majorite_proj/Majorite/Comp.majorite_top_tester(test_bench)
# do C:/EDA/ModelSim/v5_6c/ModelSim.do
# 900x300+0+0
# 1272x604+0+346
# 364x300+908+0
# hdsAddWaveButtons AddWaves
# .signals
# .wave
run —all
# ** Note: >>Debut de la simulation
#    Time: 0 ns  Iteration: 0  Instance: /majorite_top_tb/i1
# ** Note: >>Nombre d'Erreur_s détectée = 0
#    Time: 800 ns  Iteration: 0  Instance: /majorite_top_tb/i1
# ** Note: >>Fin de la simulation
#    Time: 800 ns  Iteration: 0  Instance: /majorite_top_tb/i1
```

Listing 4.2 – Informations fournies par la fenêtre Log

A titre de documentation, nous pouvons imprimer le chronogramme de la simulation comme le montre la figure 4.13

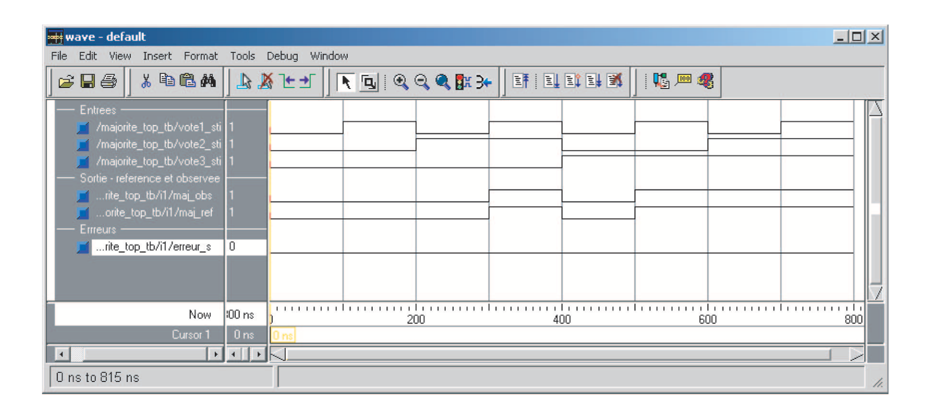


FIGURE 4.13 – Chronogramme de la simulation automatique de Majorite_top_tb

4.2.6 Redémarrage de la simulation

Pour relancer la simulation (du même module), cliquez sur le bouton ou aller dans les menus *Simulate* ▷ *Run* ▷ *Restart. . .* .

Lorsque l'on relance la simulation, les derniers fichiers **compilés** du module sont rechargés pour être simulé.

4.2.7 Impression du chronogramme

4.2.7.1 Mise en page. . .

Aller dans les menus *File* ▷ *Page setup. . .* depuis la fenêtre *Wave*.

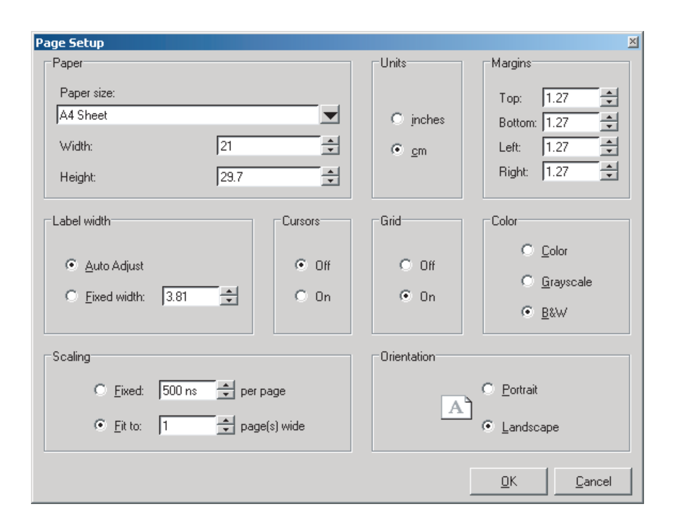


FIGURE 4.14 – Paramètres pour l'impression

Le paramètre le plus intéressant dans ce dialogue est la mise à l'échelle (*Scaling*). Il permet de spécifier selon l'option choisie :

- *Fixed ? per page*
  - la durée maximale de simulation à afficher par page.
- *Fit to ? page(s) wide*
  - le nombre de pages sur lequel le chronogramme doit être imprimé.

4.2.7.2 Impression. . .

Aller dans les menus *File* ▷ *Print* depuis la fenêtre *Wave*.

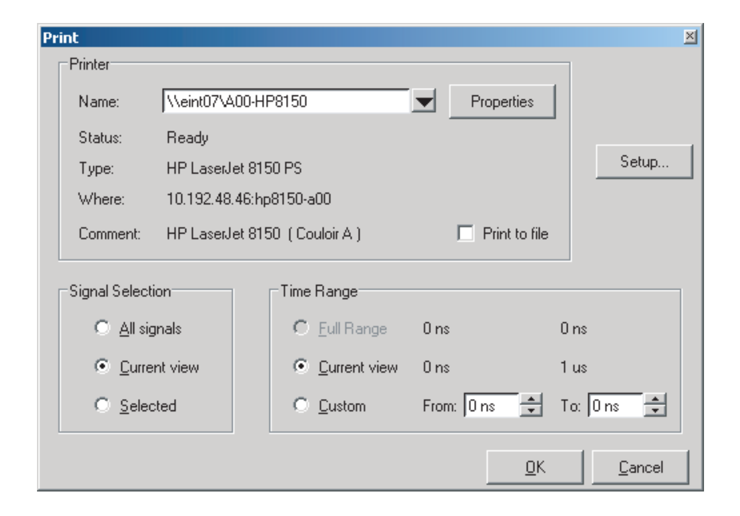


FIGURE 4.15 – Fenêtre pour l'impression

Dans ce dialogue, il est possible de spécifier quels sont les signaux du chronogramme que l'on désire imprimer :

- *All signals*
  - imprime tous les signaux .
- *Current view*
  - imprime les signaux qui sont visibles dans la fenêtre.
- *Selected*
  - imprime les signaux qui sont sélectionnés.

Il est également possible de spécifier l'intervalle de temps à imprimer :

- *Full Range*
  - imprime les traces depuis le début de la simulation jusqu'au dernier instant simuler.
- *Current view*
  - imprime les traces sur l'intervalle visible dans la fenêtre.
- *Custom*
  - imprime les traces sur l'intervalle spécifier dans les deux champs suivant.

4.2.8 Sauvegarde des traces du chronogramme

Il y a deux méthodes pour sauvegarder un chronogramme :

- Sauvegarde de la structure d'un chronogramme, soit les noms des signaux et les intercalaires présent dans la fenêtre *Wave*. Cela permet de retrouver l'environnement de travail d'une session de simulation à l'autre.
- Sauvegarde dans un fichier graphique, pour la documentation.

4.2.8.1 Sauvegarde de la structure de la fenêtre Wave

Sélectionné la fenêtre "*Wave*" en cliquant dessus.

- Enregistrement
  1. Dans les menus *File* ▷ *Save*.
  2. Choisissez le répertoir ou doit être enrgistré la configuration(*.do).

3. Vérifiez que la coche "*Waveform formats*" est mise.
4. Cliquez sur "*Ok*".

– Ouverture
1. Dans les menus *File* ▷ *Load.*
2. Sélectionné le fichier de configuration(*.do).
3. Cliquez sur "*Ouvrir*".

4.2.8.2 Sauvegarde de la fenêtre Wave dans un fichier bitmap

ModelSim permet de sauvegarder l'état présent d'un chronogramme dans un fichier graphique au format *bitmap*.

1. Sélectionné la fenêtre "*Wave*" en cliquant dessus.
2. Aller dans les menus *File* ▷ *Export.*
3. Enregistrer votre image.

4.3 Option du simulateur

4.3.1 Paramétrisation du simulateur pour les instructions "assert"

Aller dans les menus "*Simulate*" de la fenêtre principale ▷ "*Runtime Options...*".

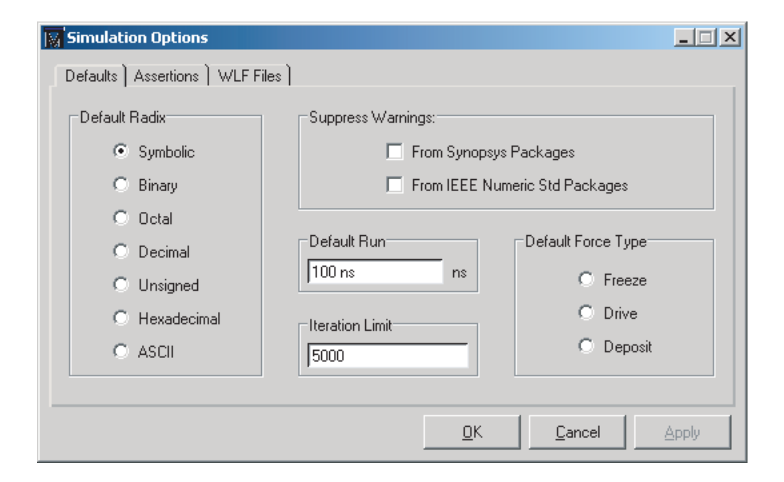


FIGURE 4.16 – Paramètres du simulateur

Sous l'onglet "*Assertions*", il est possible d'indiquer au simulateur depuis quel niveau de gravité il doit suspendre la simulation (spécifié dans la zone "*Break on Assertion*") ainsi que les instructions "*ass*ert" qu'il doit ignorer (spécifié dans la zone "*Ignore Assertions For*").

4.4 Simulation après Placement-Routage

Quartus II fournit les fichiers nécessaires pour effectuer la simulation après Placement-Routage. Il s'agit de

- *nomProjet.vho*, contenant la description.
- nomProjet_vhd.sdo, contenant les timings.

On peut le trouver à l'adresse :

<nomProjet_proj>\<nomProjet>\P_R\Simulation\modelsim

Voici les étapes pour réaliser cette simulation :

- Renommez le fichier *nomProjet.vho* en *nomProjet_pr.vhd*
- Depuis le "*Desing Explorer*", clic-droite sur le composant que l'on veut simuler. Choisir le menu *"Add"* ▷ "Gate Level".

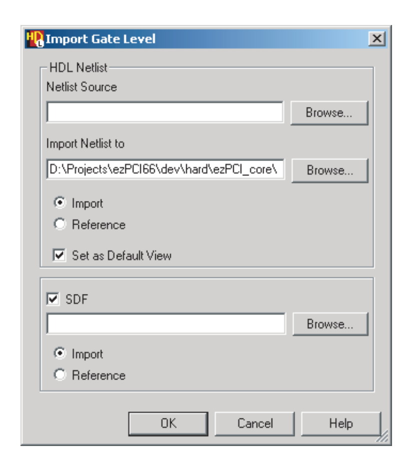


FIGURE 4.17 – Importation d'une netlist avec les contraintes de temps

- Dans la fenêtre qui s'ouvre (cf. figure 4.17)
  - sous la rubrique *"HDL Netlist"*, sélectionnez le fichier *"nomProjet_pr.vhd"*.
  - Sélectionnez l'option "*SDF*" et choisissez le fichier *"nomProjet_vhd.sdo"*.
  - Choisir deux fois l'option "*Copy the netlist file*".
- Validez avec le bouton *"OK"*.

La suite et la même que pour une simulation standart. Veuillez donc suivre les instructions depuis le § *"Simulation automatique"*.

4.5 Le banc de test

Le *Test Bench* est une entité destinée uniquement à la simulation (il n'est pas synthétisable et n'a d'ailleurs ni entrées ni sorties). Le schéma bloc d'un test bench est le suivant :

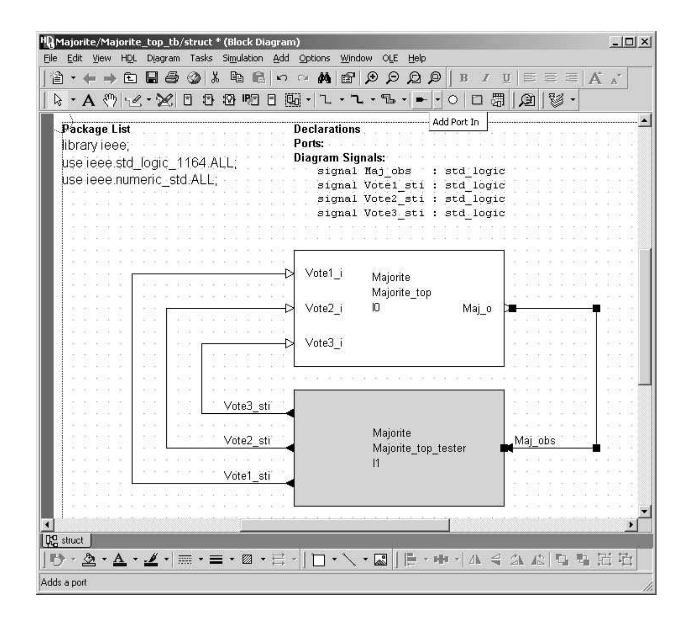


FIGURE 4.18 – Schéma bloc d'un test-bench

Le composant que l'on désire tester (*Unit Under Test, UUT*) est connecté à un bloc qui fournit les stimulus et compare les réponses avec les valeurs souhaitées (dans l'exemple *Majorite_top_tester*). Le composant et le bloc *testeur* sont reliés de manière implicite par des connexions par noms.

4.5.1 Création du Test Bench

Sélectionnez, dans le *Design Explorer*, dans la bibliothèque de travail, le composant à simuler (dans notre exemple *Majorite_top*) et aller dans les menus *"File"* ▹ *"New"* ▹ *"Test Bench..."*. Dans la fenêtre qui s'ouvre, valider en cliquant sur *"OK"*.

4.5.2 Connexion des blocs Testeur et UUT

Sélectionnez dans le *Design Manager* le module qui vient d'être créé (dans notre exemple *Majorite_top_tb*) et aller dans les menus *File* ▹ *Open* ▹ *Design content.*

Connecter les signaux ensemble soit par exemple les deux signaux *Vote1_i*, les deux signaux *Maj_o*. Puis renommer les signaux qui sortent du tester par **_sti* (pour stimuli) et ceux qui entrent dans le tester par **_obs* (pour observé). Dans notre exemple on obtient *Vote1_sti*, *Vote2_sti*, *Vote3_sti* et *Maj_obs* (cf figure 4- 19) .

Synthèse avec Precision Synthesis 5

La Synthèse est l'étape qui consiste à traduire des descriptions (du VHDL dans notre cas) en équation logique. Ce travail est réalisé par l'outil *Precision Synthesis*.

5.1 Lancement de la synthèse

- Dans le *Design Explorer* de *HDL Designer*, sélectionnez la bibliothèque où se trouve le composant à synthétiser.
- Sélectionner le composant à synthétiser, dans le cas de notre exemple *Majorite_top*.
- Lancer le synthétiseur en double-cliquant sur *"Precision Synthesis Flow REDS"* dans la fenêtre *Tasks*.
  - Le fichier VHDL contenant l'ensemble des fichers VHDL hiérarchiques correspondant au composant sélectionné (concaténation) , *NomComposant_concat.vhd* sera créé automatiquement.

Presicion Synthesis se lance et charge automatiquement les fichiers à synthétiser.

5.2 Interface de precision

Une fois le programme lancé, on obtient la fenêtre representée à la figure 5.1. Cette fenêtre est divisée en deux, le *Design Bar* sur la gauche et une zone d'affichage qui peut être divisée par un certain nombre d'onglet.

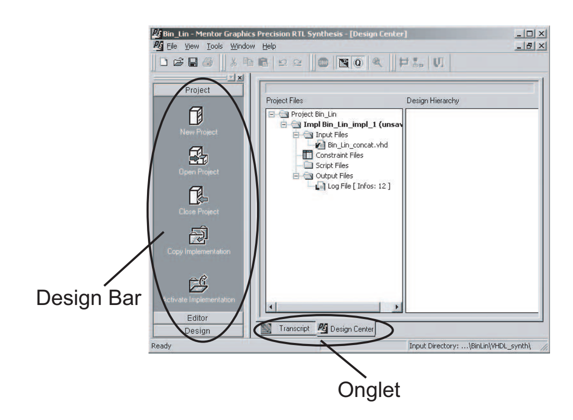


FIGURE 5.1 – Fenêtre initiale obtenue après le lancement de *Precision Synthesis*

Description des onglets :

- *Design Bar*
  - Regroupe les différentes commandes disponibles dans *Precision*. Elles sont rangées en différents *tiroirs*. Ces commandes ne sont disponibles qu'au fur et à mesure que l'on avance dans le projet.
- Onglet

– *Transcript Zone* où sont affichées les informations données par *Precision* (déroulement du process, Warning, erreurs, etc ...).
– *Design Centre Zone* où sont regroupées tous les fichiers d'entrées ou de sorties de *Precision*. C'est également là qu'il est possible d'affecter différentes contraintes au projet.
– D'autres onglets apparaissent pour afficher les informations demandées, par exemple pour afficher les rapports (timing ou Aera) ou pour afficher une vue (*RTL* ou *View Technology Schematic*).

5.3 Synthèse de la description

5.3.1 Choix de la cible

1. Dans la *Design Bar*, sélectionnez le tiroir *Design*.
2. Cliquez sur l'icon *Setup Design* la figure 5.2 apparaît.
3. Configuration de la cible,

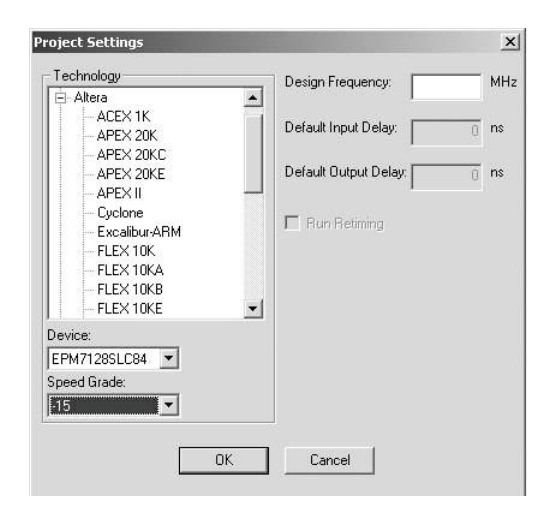


FIGURE 5.2 – Sélection de la cible

Remarque : Le tableau ci-dessous référence les informations sur les circuits les plus couramment utilisés au laboratoire. Vérifiez quelle est la cible utilisée !

Technologie	Device	Speed grade
	EPM7064SLC44	-10 ou -15
Altera MAX 7000S	EPM7128SLC84	-7 ou -15
Altera Acex 1K	EP1K30QC208	-3

TABLE 5.1 – Liste des circuits les plus fréquemment utilisés

Le choix de la technologie permet :

– d'orienter *"Precision"*dans sa manière de simplifier les "équations logique" (simplification au niveau des portes logiques).
– de connaître le circuit cible voulu afin de le transmettre au *placeur-routeur*. Ceci évite de devoir le spécifier manuellement lors du placement-routage.

5.3.2 Compilation

Toujours dans le menu *Design* du *Design Bar*, lancez la compilation en cliquant sur le bouton *Compile*.

Durant cette opération, une analyse est effectuée sur les descriptions VHDL afin les convertir en un schéma fonctionnel équivalent (fonction logique standard). Il est possible de visualiser ce schéma en sélectionnant le menu *Tools* ▷ *View* ▷ *View RTL Schematic*.

Remarque : *Precision* permet d'afficher les schémas dans une ou plusieurs vue. Pour passer d'un mode à l'autre, faire un click droite sur le graphique et activer (ou désactiver) l'option *"MultiPage Schematics"*.

5.3.3 Affectation des broches

Aller dans les menus *File* ▷ *Run Script...* puis ouvrir le fichier d'assignation des *pins* aux *ports* (extension *"*.tcl"*).

5.3.4 Synthesize

Toujours dans le menu *Design* du *Design Bar*, lancez la *synthèse* en cliquant sur le bouton *Synthesize*.

Durant cette phase, *Precision Synthesis* va essayer de simplifier le schéma de portes équivalent afin de minimiser soit la place utilisée dans le silicium, soit le temps de propagation entre les entrées et les sorties.

Il est possible de voir le schéma de portes logiques optimisé pour la technologie sélectionnée en choisissant dans le menu *Tools* ▷ *View* ▷ *View Technology Schematic*. Durant cette phase, *Precision Synthesis* crée aussi les différents fichiers nécessaires pour le *placement-Routeur Quartus II*.

5.4 Fichier d'assignations de "pin"

Le fichier utilisé pour l'assignation des *pins* est un fichier *TCL* ("*Tool Command Language*"). Ce fichier est un script effectuant une suite de commandes dans "*Precision Synthesis*".

5.4.1 Script GenePin.tcl pour carte EPM 25p - 25p

Le script *GenePin.tcl* permet de générer automatiquement ce fichier d'affectation des pins pour la carte *EPM 25p - 25p* disponible au laboratoire.

FIGURE 5.3 – Interface du script GenePin.tcl

5.4.2 Syntaxe d'une assignation

Ce fichier est composé d'une suite d'assignation. Il est possible d'ajouter des commentaires en le faisant précéder du caractère "#".

set_attribute Nom_Signal -name PIN_NUMBER -value Numero_Pin -port

où il faut remplacer :

- *Nom_Signal* Nom du signal auquel le numéro de "*pin*" doit être associé. Pour les vecteurs, ce nom est suivit (sans espace) de l'indice du signal entre parenthèses.
- *Numero_Pin* Numéro de "*pin*" à associer au signal "*Nom_Signal*".

```
# Fichier : Afficheur.tcl
# ...

# Port d'entree A
set_attribute a(4) -name PIN_NUMBER -value  4 -port
set_attribute a(3) -name PIN_NUMBER -value  5 -port
set_attribute a(2) -name PIN_NUMBER -value  6 -port
set_attribute a(1) -name PIN_NUMBER -value  7 -port
set_attribute a(0) -name PIN_NUMBER -value  8 -port

# Port d'entree B
set_attribute b(4) -name PIN_NUMBER -value  9 -port
set_attribute b(3) -name PIN_NUMBER -value 11 -port
set_attribute b(2) -name PIN_NUMBER -value 12 -port
set_attribute b(1) -name PIN_NUMBER -value 13 -port
set_attribute b(0) -name PIN_NUMBER -value 14 -port

# Port de sortie C
set_attribute s(6) -name PIN_NUMBER -value 18 -port
set_attribute s(5) -name PIN_NUMBER -value 19 -port
set_attribute s(4) -name PIN_NUMBER -value 20 -port
set_attribute s(3) -name PIN_NUMBER -value 21 -port
set_attribute s(2) -name PIN_NUMBER -value 24 -port
set_attribute s(1) -name PIN_NUMBER -value 25 -port
set_attribute s(0) -name PIN_NUMBER -value 26 -port
```

Listing 5.1 – Fichiers d'assignation des numéros de pins aux ports

6 Placement-Routage et intégration avec Quartus II

Pour lancer le logiciel, allez dans les menus :

Démarrer ▷ *Labo numérique* ▷ *EDA* ▷ *Quartus II 7_2*

6.1 Ouverture du projet

Pour créer un nouveau projet, cliquez sur le bouton, une fenêtre s'ouvre. Sélectionnez le répertoire *P_R* de la bibliothèque contenant le composant (répertoire ..*Majorite_proj**Majorite**P_R*) et valider en cliquant sur le bouton *Créer*.

Quartus II crée le projet et ajoute à ce dernier le fichier source (**.edf*) ainsi que les contraintes spécifiées lors de la synthèse dans *Precision Synthesis*.

Remarque : Un bouton a été ajouté dans Quartus : . Ce bouton est utilisé pour rependre un projet après la *synthèse* de ce dernier par *Precison*.

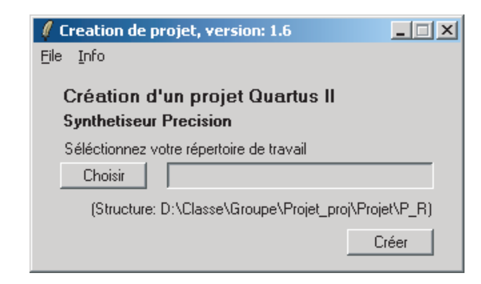


FIGURE 6.1 – Création d'un nouveau projet

6.2 Placement et routage

Affichez la fenêtre de compilation en allant dans les menus *Quartus II* ▷ *Compiler Tool*. La fenêtre *Compiler Tool* apparaît (cf. figure 6.2).

Lancer le *placement-routage* en cliquant sur le bouton "> *Start*".

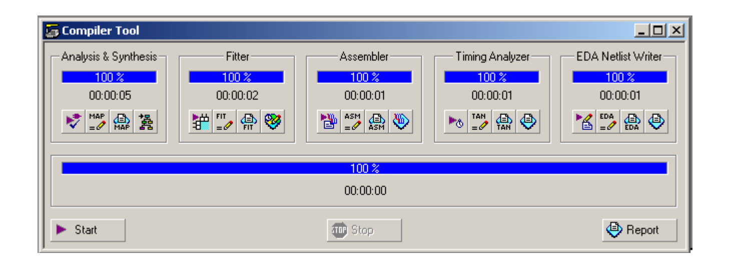


FIGURE 6.2 – Compiler Tool

6.2.1 Vue RTL

Quartus II permet de voir une vue *RTL* du projet. Aller dans les menus *Quartus II* ▷ *RTL Viewer*.

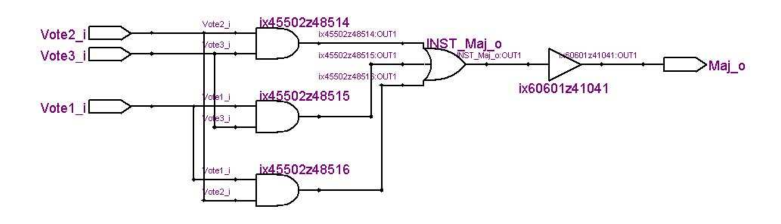


FIGURE 6.3 – Vue RTL du projet Majorite

6.2.2 Visualisation des rapports

Pour visualiser les résultats obtenus lors du placement-routage, *Quartus II* crée un certain nombre de fichiers rapports. Pour les visualiser, cliquer sur le bouton *Report* dans la fenêtre *Compiler Tool*.

6.2.3 Fichier générer

Quartus II génére une grande quantité de fichiers, voici les plus importants :

– **.pof/*.sof* fichier de configuration du circuit programmable.
– **.fit.rpt* résumé du résulat de placement-routage.
– **.fit.eqn* fourni les équations logiques obtenues.
– **.pin* Liste avec l'affectation des broches.

6.3 Programmation

6.3.1 Branchement de la carte

La programmation se fait toujours avec un cable de type *ByteBlaster*.

Pour la connexion, branchez le câble dans le port imprimante du côté PC et dans le connecteur prevu à cette effet sur la carte.

Remarque : Pour chacune des cartes disponibles au laboratoire une documentation a été réalisée. Veuillez-vous y référer en cas de doute !

6.3.2 Lancement du programmateur

Pour lancer le programmeur, aller dans les menus *Quartus II* ▷ *Programmer*. La fenêtre de programmation apparaît (cf. figure 6.5).

6.3.3 Sélection du "module" de programmation

Le "*module*" de programmation est un *ByteBlaster*. Vérifiez que le module *ByteBlaser* est bien bien sélectionné (cf. figure 6.4), si se n'est pas le cas :

- Cliquez sur le bouton *Hardware Setup*.
- Dans la fenêtre qui apparaît, sélectionner le *ByteBlster* dans la zone *AvailableHardware items*.
- Sur la ligne *Currently Selected Hardware*, vous devriez avoir *ByteBlaster [LPT1]*.
- Validez avec le bouton *Close*.

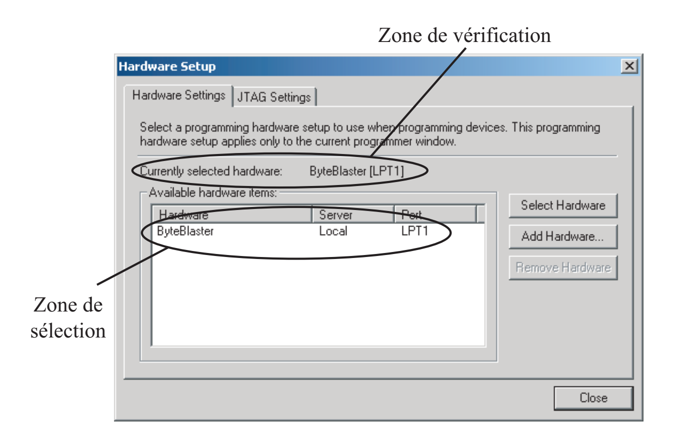


FIGURE 6.4 – Sélection du module de programmation

6.3.4 Programmation de la carte

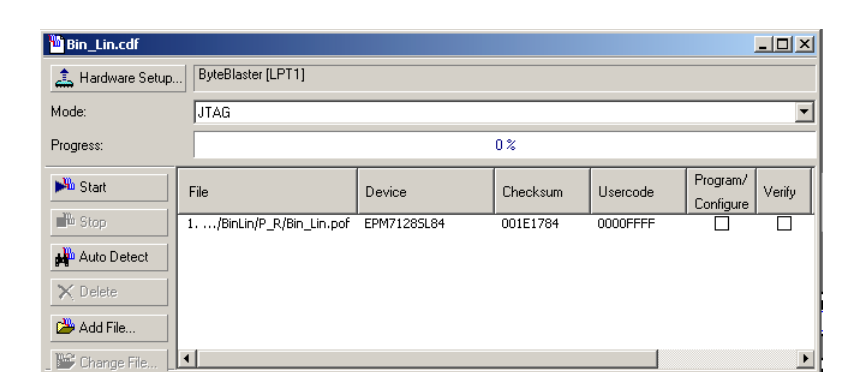


FIGURE 6.5 – Fenêtre de programmation

- Cochez les cases *Program/configure* et *verify* (cf. figure 6.5).
- Lancez la programmation en cliquant sur le bouton *Start*.

6.3.5 Reprise d'un projet

Pour programmer un circuit à partir d'un fichier de programmation disponible, suivez les étapes suivantes :

- Lancez le programme *Quartus II*.

- Ouvrir le *programmeur*, avec le menu *Quartus II* ▷ *Programmer*.
- Sélectionnez le fichier de configurations en cliquant sur le bouton *Add File...*. Les fichiers de configurations ont comme extansion **.pof (Max7000S)* ou **.sof (Acex)*.
- Suivre les informations du § 6.3, "*Programmation*"



A.1 Convention de noms au REDS

La convention de noms utilisée pour les indentificateurs est la suivante :

- 1ère lettre en majuscule, le reste en minuscule.
- Chaque mot est séparé par un souligné avec la 1ère lettre en majuscule.
- Voici quelques exemples :
  - Bus_Donnee, Etat_Present, Adr_Sel, Val_Don.

Afin de simplifier la lecture des descriptions VHDL, nous avons défini une convention pour les noms des signaux. Les tableaux ci-après vous donnent les suffixes et préfixes utilisés :

Objets	**Suffixes**
constante	_c
variable	_v
port entrée	_i
port sortie	_o
port entrée/sortie	_io
signal interne :	
architecture textuelle VHDL	_s
architecture interne (schéma bloc)	-
spécifique pour banc de test (test-bench) :	
signaux de stimuli	_sti
signaux observés	_obs
signaux de référence	_ref

Table A.1 – Liste des suffixes utilisés au REDS

Polarités	Préfixe	Exemple
signaux actif haut	-	Up_i
signaux actif bas	'n'	nUp_i
signaux double	'n' devant second nom	Up_nDn_i

Table A.2 – Liste des préfixes utilisés au REDS

A.2 Résumés des flows de simulation

A.2.1 Simulation manuelle avec la console

- Depuis le *Design Manager*, cliquer sur le *Top_Sim* pour ouvrir la vue structurelle (schéma).
- Ajouter et Connecter le composant à simuler dans le *Top_Sim*.

- Vérifiez que l'architecture à simuler est celle sélectionnée par défaut dans le *Design Manager.*
- Depuis le *Design Manager*, sélectionnez le composant *Top_Sim*.
- Cliquez sur *Compile* dans la fenêtre *Tasks* (en haut à gauche dans le *Design Manager*).
- Cliquez sur *Simulate* dans la fenêtre *Tasks.*
- Dans la fenêtre qui apparaît, validez en cliquant sur *OK*. Le simulateur *ModelSim* se lance.
- Dans la fenêtre de *ModelSim*, cliquez sur le bouton *REDS_console* pour ouvrir celle-ci.
- Ajouter les signaux du composant à simuler (pas ceux de *Top_Sim*) dans la fenêtre *Wave.*
- Choisir l'état des entrées désirées dans la console (interrupteur ou valeur).
- Activer un pas de simulation en cliquant sur le bouton *RUN.*
- Répéter ces 2 derniers points pour toutes les combinaisons à tester.

A.2.2 Simulation automatique

- Vérifiez que l'architecture à simuler est celle sélectionnée par défaut dans le *Design Manager.*
- Depuis le *Design Manager*, sélectionnez le *test-bench* du composant à simuler, portant le nom *NomComposant_tb.*
- Cliquez sur *Compile* dans la fenêtre *Tasks* (en haut à gauche dans le *Design Manager*).
- Cliquez sur *Simulate* dans la fenêtre *Tasks.*
- Dans la fenêtre qui apparaît, validez en cliquant sur *OK*. Le simulateur *ModelSim* se lance.
- Ajoutez les signaux dans la fenêtre *Wave.*
- Lancez la simulation